University of Alberta

Phase Transitions and Typical-case Complexity: Easy (Hard) Aspects of Hard (Easy) Problems

by

Yong Gao

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2005

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

0-494-08642-4

Your file *Votre référence*
ISBN
Our file *Notre reference*
ISBN

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

In this thesis, we study theoretically and empirically the typical-case hardness of randomly-generated instances of several algorithmic problems that are of interest in artificial intelligence research. For randomly-generated instances of constraint satisfaction problems (CSP), we identified a new class of algorithmically exploitable structures and proved that under certain instance distributions, random instances contain such structures with high probability (Chapter 4). In an effort to find a way to eliminate these structures from randomly-generated CSP instances, we established an interesting connection between the notion of constraint consistency in the literature and the resolution complexity of random CSP instances. By embedding a recursive structure called consistency core into random CSP models, we proposed a novel scheme to generate random CSP instances with theoretically guaranteed resolution complexity and empirically confirmed hardness (Chapter 5). Our proposal resolved the long-standing problem of generating hard random CSP instances with bounded domain size that has troubled the society for several years.

While all of the results in Chapters 4 and 5 are aimed at backtracking search algorithms, we investigated in Chapter 6 the typical-case behavior of random instances in terms of the dynamic programming algorithms whose time and space complexities are exponential in the treewidth of the underlying structures. This type of algorithm has been widely used in the study of Bayesian network inference and CSPs. We established an improved lower bound on the threshold for a random graph to have a treewidth linear in the graph size. Similar techniques were then applied to random CSPs, random Bayesian networks, and fitness landscape models in computational biology and evolutionary computation.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Joseph Culberson, for his advice, inspiration, and encouragement over the years. I thank his patience while I was at a loss in my first year as an international student in 1998, his understanding when I decided to graduate with an MSc in 2000, and in particularly his encouragement when I was exploring the idea of "going back to school" to do a PhD in 2002. It is no exaggeration to say that without him, this thesis would never have been written, or even have been started.

I also want to thank my other committee members, Toby Walsh, Mohammad R. Salavatipour, Lorna Stewart, Yau Shu Wong, and Russell Greiner, for reading my proposal and thesis draft. Their suggestions and criticisms during my candidacy exam are an invaluable source of inspiration during the preparation of this thesis.

I am grateful to many people with this department, in particularly to Dr. Peter van Beek (now with Univ. of Waterloo), Dr. Lorna Stewart, Dr. Guohui Lin, Dr. Kui Wu (now with Univ. of Victoria), and (soon to be) Dr. Calin Anton and Yuxi Li. Studying and working with them gives me an opportunity to know a variety of research fields and research perspectives which otherwise will take forever for me to learn by myself. Thanks are also devoted to Dr. David G. Mitchell of Simon Fraser University who kindly provided me a copy of his thesis.

Finally, I appreciate my family and parents for their patience and support through good times as well as bad times.

Contents

1 Introduction 1

2 Preliminaries 5

2.1 The Probabilistic Method 6

2.2 Theory of Random Graphs 8

2.3 Decision Problems, Computational Complexity, and Proof Complexity 10

2.3.1 Resolution Proof System 11

2.3.2 Hard Formulas for Resolution 12

2.3.3 DPLL Procedure and Resolution Complexity 15

2.4 Phase Transitions and Typical-case Complexity 17

2.4.1 SAT 18

2.4.2 Graph Coloring 22

2.4.3 Hamiltonian Cycle 23

2.4.4 Number Partitioning 24

2.5 Generating Hard Instances 27

3 Constraint Satisfaction Problem and its Random Models 31

3.1 Constraint Satisfaction Problem 31

3.2 Random Models of Constraint Satisfaction Problems 34

3.2.1 Classical Random CSP Models 36

3.2.2 Improved Random CSP Models 37

3.3 Phase Transitions of Random CSPs 38

3.3.1 Flawed Variables, Flawed Constraints, and Well-behaved CSPs 38

3.3.2 Sharpness of CSP Phase Transitions 41
3.3.3 Random CSPs with (Slowly) Increasing Domain Size . 41

4 Random CSPs with Polynomial Resolution Complexity 43
4.1 Introduction 43
4.2 Main Results 44
4.3 Proofs of the Results 46
4.3.1 Proof of Theorem 4.2.1 46
4.3.2 Proof of Theorem 4.2.3 55
4.4 Discussions 56

5 Consistency and Better Random CSP Models 57
5.1 Introduction 57
5.2 Consistency and Resolution Complexity of Random CSPs . . . 58
5.2.1 CNF Encoding of CSPs 59
5.2.2 Consistency and Resolution Complexity of Random CSPs 61
5.3 Consistency Core and Harder Random CSP Model with High Constraint Tightness 61
5.3.1 Flawless Random CSPs 62
5.3.2 Generalized Flawless Model and Consistency Core . . . 63
5.4 Experiments 66
5.4.1 Effect of an Increase in Constraint Tightness 66
5.4.2 Comparisons between Three Random CSP Models . . 70
5.5 Proof of the Theorems 76
5.5.1 Theorem 5.3.1 76
5.5.2 Theorems 5.2.1 and 5.2.2 79

6 Easy Random Problems Are Sometimes Hard 85
6.1 Introduction 85
6.2 Notation and Definitions 86
6.3 Threshold of Linear Treewidth in Random Graphs 88
6.4 Treewidth of Random Models in AI and Computational Biology 95
6.4.1 Treewidth of Random CSPs 95

6.4.2 Treewidth of Random Bayesian Networks 96
6.4.3 Treewidth of NK Landscapes and Other Additive Fitness Functions 100

7 Conclusions **103**

Bibliography **107**

List of Tables

5.1 Maximum Median Number of Branches of zChaff on random instances of three random CSP models , over all $\frac{m}{n}$. Domain size $d = 4$ and $\mathcal{K}$ is 2-regular. 71

5.2 Median Number of Branches of zChaff on random instances of three random CSP models at the smallest $\frac{m}{n}$ where the solution probability is less than 0.1. Domain size $d = 4$ and $\mathcal{K}$ is 2-regular. 71

5.3 Median Number of Branches of zChaff on random instances of three random CSP models at the largest $\frac{m}{n}$ where the solution probability is greater than 0.9. Domain size $d = 4$ and $\mathcal{K}$ is 2-regular. 71

5.4 Median number of branches (median time in seconds) of ZChaff and Satz on two random CSP models with n = 500, d = 4, and t = 6. 100 instances for each parameter. 72

List of Figures

4.1 The upper bound $u(t)$ for the threshold $c_3(t)$ as a function of tightness t. Left figure: the function itself. Right figure the derivative of the function. 46

4.2 An illustration of a k-cc-loop. Only the cyclic variables $v_i, 1 \leq i \leq 3p$, are shown. Each hyper-edge E_i contains two cyclic variables from V and $(k-2)$ variables from $X \setminus V$. 47

4.3 An illustration of a set of $q = 6$ shared hyper-edges that form a hyper-tree containing three hyper-path branches. The variable v appears in 3 hyper-edges. There are 4 fixed cyclic variables, 3 limited cyclic variables. 53

5.1 A special type of consistency core with the domain size 9 . . . 66

5.2 Thresholds for the solution probability in the model $\mathcal{F}^{3,a}_{n,m}$ with $n = 250$. The z-axis is the solution probability. The axis with the range 1—2 is for the parameter $1 + a$ and the axis with the range 1—6 is for the clause density m/n. 68

5.3 Effects of an increase in the constraint tightness on the instance hardness for $\mathcal{F}^{3,a}_{n,m}$ with $n = 250$. The z-axis is the median number of branches in log-scale. The axis with the range 1.2—1.8 is for the parameter $1+a$ and the axis with the range 2.5—5.5 is for the clause density m/n. 69

5.4 Solution probability thresholds for the three random CSP models with $n = 500, t = 6$. For the generalized flawless model, $\mathcal{K}$ is set to be 2-regular. The y-axis is solution probability and x-axis is the constraints-variables ratio m/n. Sample size for each data point is 100. 72

5.5 Hardness for the three random CSP models with $n = 500, t = 6$. For the generalized flawless model, $\mathcal{K}$ is set to be 2-regular. The y-axis is the median number of branches used by zChaff and x-axis is the constraints-variables ratio m/n. Sample size for each data point is 100. 73

5.6 A closeup at the region $m/n = 1.8 - - - 2.5$ for the generalized flawless model with $n = 500, t = 6$ and $\mathcal{K}$ being 2-regular. Sample size for each data point is 200. Two curves are plotted. One is the median number of branches for satisfiable sample instances only, another is the average number of branches for all the sample instances. 74

7.1 These curves are all supposed to drop to zero in the limit. . . 106

Chapter 1

Introduction

In physics, the notion of a *phase transition* refers to the abrupt change of states (phases) of compounds at some values of the system parameters such as the pressure and the temperature. Popular examples of phase transitions in the physical world are the liquid-to-gas transition of water, and the conductor-superconductor transition of electrical resistance of some materials.

Similar phenomena have also been observed in computational and artificial intelligence (AI) systems. For example, consider a random graph on n vertices generated by selecting each of the $n(n-1)/2$ potential edges independently with edge probability $p = p(n)$. It is well-known that the random graph experiences several abrupt changes in its combinatorial properties when the edge probability $p(n)$ increases from $o(1/n)$ to $\Theta(1)$ [30, 59]. For many NP-complete problems such as the Boolean Satisfiability problem (SAT) and the graph coloring problem, the probability for a randomly-generated instance to have a solution also has a phase transition from zero to one at a certain value of the parameter that controls the "density" of the randomly-generated instances [35, 41, 101]. In their seminal work [35], Cheeseman at al. showed that for many NP-complete problems and some standard search algorithms, the typical hardness of randomly-generated instances is closely related to the critical point, called the *threshold*, where the phase transition occurs.

There are several reasons why a study of phase transitions in NP-complete problems is interesting. First, while many algorithmic problems of great importance in AI and other practical fields are NP-complete, it is not necessarily true that instances of these problems are equally hard. People are particularly interested in a theory that addresses the "typical-case" complexity of these problems and helps determine the regions in the problem space where instances are relatively easy to solve. Second, in the empirical analysis of algorithms, the selection of reasonable benchmarks is an important factor. In addition to real-world benchmarks, random problem instances are widely used. However, generating really hard random instances is not a trivial task. A classical example is a random SAT model used in early 1980s that has been shown to have an extremely strong bias towards generating, sometimes triv-

ially, easy instances [78, 62]. A more recent example is a widely used model for the constraint satisfaction problem (CSP) which has been proved to be trivially unsatisfiable asymptotically with probability one [8]. An analysis of the typical-case complexity of randomly-generated instances of NP-complete problems may thus provide a valuable guidance to the design of random instance generators [6, 74].

Recent studies on the phase transitions in NP-complete problems have given us much insight into the typical-case complexity of these problems and help in tackling questions such as "where are the really hard problems?" and "why do these hardest problems seem to resist any intelligent algorithms?" [39, 44, 63, 114]. Answers to these questions have already stimulated research on designing efficient algorithms and appropriate benchmarks [6, 109, 118]. See also [85, 86, 87, 118] for a series of popular science articles for the history and recent development.

A lesson learned from the study of the phase transition in NP-complete problems is the central importance of the structural information in a combinatorial search problem. We now have a clearer view regarding why really hard problems "are well out of reach of any intelligent algorithms"—there is simply no small structural signatures in these hard problems for any foreseeable intelligent algorithms to exploit [44]; Random models with richer structures have been proposed to avoid the triviality in existing random CSP models and to generate better testing instances in the study of search algorithms [6, 8, 74, 79]; Combinatorial search problems have also been investigated on some non-classical models of random graphs such as the power-law graphs and the small-world graphs [136, 137]. These graphs have unique structural characteristics and have been found to be ubiquitous in communication networks, biological systems, and human natural languages [13].

It is commonly believed that the easy-hard complexity pattern associated with the phase transition of the solution probability is algorithm-independent. While this is true for the class of backtracking search algorithms, there has been evidence showing that algorithms making use of different structural information may have very different behaviors [38].

This thesis, as its title suggests, contributes to the ongoing research on the phase transitions and typical-case complexity by investigating easy aspects in the region of the problem space where random instances have been expected to be difficult to solve, and hard aspects for some type of search algorithms in the region of the problem space where random instances have been proven to be typically easy for backtracking algorithms.

Contributions

Polynomial Resolution Complexity of Random CSPs

For some popular random CSP models, we identify a new class of subproblem structures whose appearance makes randomly-generated CSP instances trivially easy. We prove that this type of subproblem appears asymptotically with probability one in a region of the problem space where instances are free of the triviality demonstrated by Achlioptas et al. [8]. This result partly answers an open question regarding the (resolution) complexity of random CSPs posed in [113].

Consistency, Resolution Complexity, and Better Random Models of CSPs

Since the work of Achlioptas et al. [8] on the triviality of random CSP models, there has been much effort in designing better random CSP models that exhibit non-trivial threshold behaviors and have guaranteed hard instances at phase transitions [8, 42, 75, 113, 115, 129, 140]. One of the most significant problems with these random CSP models is that as a model parameter, the constraint tightness has to be very low for bounded domain size. We establish an interesting connection between the notion of constraint consistency in the literature and the resolution complexity of random CSP instances. By embedding a recursive structure called consistency core into random CSP models, we propose a novel scheme to generate random CSP instances with theoretically guaranteed resolution complexity and empirically confirmed hardness.

Typical Size of Treewidth of Random Graphs and Other Random Structures

We study the typical size of the treewidth of random graphs and graph structures of some randomly-generated problem instances. The significance of the typical size of the treewidth lies in the fact that the time and space complexities of many popular non-backtracking search algorithms are exponential in the treewidth of the underlying graph structures of the problems, including the tree-decomposition-based algorithms for CSPs, some exact inference algorithms for Bayesian networks, and the *estimation of distribution algorithms* in evolutionary computation.

We establish an improved lower bound on the threshold for a random graph to have a linear treewidth. Using the same analytical technique, we further show that the graph structures associated with randomly-generated instances of CSPs, Bayesian networks, and fitness landscapes all have a treewidth linear in the problem size, even in the region of the problem space where backtracking algorithms have been shown to be very efficient.

Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 introduces the notation, concepts, and techniques from the probabilistic method, the theory of random graphs, and the theory of computational complexity. Some of them will be used in the remaining chapters of the thesis. The last two sections of this chapter overview the study of the phase transitions and typical-case complexity, including the phase transitions of four typical NP-complete problems (Section 2.4), and previous studies on generating hard random instances (Section 2.5). In Chapter 3, we introduce the constraint satisfaction problem, its random models, and existing work on its phase transitions.

In Chapter 4, we establish a set of lower bounds on the constraint tightness, an important parameter of random CSP models. Above these lower bounds, randomly-generated CSP instances have a polynomial resolution complexity asymptotically with probability one.

In Chapter 5, we prove some theoretical results on the connections between constraint consistency, another important concept intensively discussed in the CSP literature, and the resolution complexity of randomly-generated CSP instances. Based on these connections, we propose a novel scheme that can be used to design new random CSP models to overcome the difficulties with the classical random models. A series of empirical results are also reported on the relation between the constraint consistency and the resolution complexity as well as on the comparison between our proposed random CSP model and previous models.

Chapter 6 is devoted to a discussion on the hard aspect of (typically) easy ensemble of problem instances. First, we establish an improved lower bound on the threshold for a random graph to have a linear treewidth. Then using similar analytical techniques, we show that the typical size of the treewidth of the underlying graph structures is also large in random CSPs, random Bayesian networks, and some other models in computational biology and evolutionary computation. The obtained results indicate that several algorithms developed in the CSP and Bayesian network communities have a typically exponential behavior in the region of the problem space where randomly-generated instances can be solved easily by backtracking algorithms. Chapter 7 is the conclusion.

Chapter 2

Preliminaries

Let $\mathcal{P} = (\Omega, \mathcal{A}, \Pr)$ be a probability space where Ω is a sample space, $\mathcal{A}$ is a σ-field, and Pr is a probability measure. Throughout the thesis, we will use the following notations:

$\mathcal{E}_{\mathcal{P}}[X]$: the expectation of a random variable X;
$\sigma^2_{\mathcal{P}}[X]$: the variance of a random variable X;
I_A : the indicator function of an event $A \in \mathcal{A}$.

When the probability space is clear from the context, we will suppress the subscripts and simply write $\mathcal{E}[X], \sigma^2[X]$, and I_A.

Let $\{\mathcal{P}_n = (\Omega_n, \mathcal{A}_n, \Pr_n), n \geq 1\}$ be a sequence of probability spaces and let $\{A_n \in \mathcal{A}_n, n \geq 1\}$ be a sequence of events. We say that $\{A_n \in \mathcal{A}_n, n \geq 1\}$ occur *with high probability* (**whp**) if $\lim_n \Pr_n\{A_n\} = 1$.

A random variable X has the *Bernoulli distribution* with parameter $p \in [0,1]$ if $\Pr\{X = a\} = p^a(1-p)^{1-a}, a \in \{0,1\}$. The sum of a sequence of n independent Bernoulli random variables has the *binomial distribution* $b(n,p)$ with parameters n and p. The following Stirling's formula and related inequalities for the binomial coefficients are also frequently used:

$$n! = \left(\frac{n}{e}\right)^n \sqrt{2\pi n} e^{\theta}, 0 < \theta < 1 \tag{2.1}$$

$$c_1\sqrt{\delta(1-\delta)n}\left(\frac{1}{\delta^{\delta}(1-\delta)^{1-\delta}}\right)^n \leq \binom{n}{\delta n}$$
$$\leq c_2\sqrt{\delta(1-\delta)n}\left(\frac{1}{\delta^{\delta}(1-\delta)^{1-\delta}}\right)^n \tag{2.2}$$

where δ, c_1, and c_2 are fixed constants.

2.1 The Probabilistic Method

The *probabilistic method*, initiated by Erdös and Renyi in their work on random graphs [59], is a powerful technique that uses probabilistic arguments to tackle problems of combinatorial nature. Over the years, the probabilistic method has found various applications in many branches of theoretical computer science and discrete mathematics. Probabilistic tools crucial to the use of the probabilistic method include basic inequalities such as Markov's inequality and Chebyshov's inequality, as well as more advanced results on Poisson approximations and large deviation bounds of random variables [14, 30, 131].

The First Moment Method

In the *first moment method*, the probability of some event of interest is bounded by the expectation of the corresponding random variable. Markov's inequality provides a convenient way to establish such a bound.

Lemma 2.1.1. *Let X be a random variable, $k \geq 1$ be an integer. Then, for any $t > 0$,*

$$\Pr\{|X| > t\} \leq \frac{\mathcal{E}[|X|^k]}{t^k}. \tag{2.3}$$

Proof.

$$\begin{aligned}
\mathcal{E}[|X|^k] &= \mathcal{E}[|X|^k I_{\{|X| \leq t\}}] + \mathcal{E}[|X|^k I_{\{|X| > t\}}] \\
&\geq \mathcal{E}[|X|^k I_{\{|X| > t\}}] \geq \mathcal{E}[t^k I_{\{|X| > t\}}] \\
&= t^k \Pr\{|X| > t\}.
\end{aligned}$$

□

Corollary 2.1.1 (Markov's Inequality). *Let X be a positive random variable. Then, for any $t > 0$,*

$$\Pr\{X > t\mathcal{E}[X]\} \leq \frac{1}{t}. \tag{2.4}$$

In particularly, for any positive integer-valued random variable X,

$$\Pr\{X > 0\} \leq \mathcal{E}[X]. \tag{2.5}$$

The Second Moment Method

The *second moment method* is typically used to bound the probability of the event that a random variable is within a specific interval around its expectation. The bound is based on inequalities that involve the variance of the random variable.

Lemma 2.1.2 (Chebyshov's Inequality). *Let X be a random variable. For any $t > 0$,*

$$\Pr\{|X - \mathcal{E}[X]| \geq t\mathcal{E}[X]\} \leq \frac{\sigma^2[X]}{t^2\mathcal{E}^2[X]}. \tag{2.6}$$

In particular, if $\mathcal{E}[X] > 0$, we have

$$\Pr\{X = 0\} \leq \frac{\sigma^2[X]}{\mathcal{E}^2[X]} \tag{2.7}$$

Another way to bound the probability $\Pr\{X = 0\}$ is to use the Cauchy-Schwarz inequality

$$\mathcal{E}^2[XY] \leq \mathcal{E}[X^2]\mathcal{E}[Y^2]$$

for two random variables X and Y.

Lemma 2.1.3. *Let X be a random variable. We have*

$$\Pr\{X = 0\} \leq 1 - \frac{\mathcal{E}^2[X]}{\mathcal{E}[X^2]}. \tag{2.8}$$

Proof.

$$\begin{aligned} \mathcal{E}^2[X] &= \mathcal{E}^2[XI_{\{X\neq 0\}}] \leq \mathcal{E}[X^2]\mathcal{E}[I_{\{X\neq 0\}}] \\ &= (1 - \Pr\{X = 0\})\mathcal{E}[X^2]. \end{aligned}$$

□

The Bounded Differences Method

Sometimes it is desirable to have sharper bounds on the tail probability of random variables, i.e., bounds that decreases exponentially fast. The *bounded differences method* provide such exponential bounds for "smooth" functions of random variables.

The classic exponential bounds on large deviations is the Chernoff bound for binomial random variables, i.e., the sum of independent and identically distributed Bernoulli random variables.

Lemma 2.1.4 (Chernoff Bound [110]). *Let $\{X_1, \ldots, X_n\}$ be a sequence of independent and identically distributed Bernoulli random variables with $\Pr\{X_i = 1\} = p$ for each $1 \leq i \leq n$. Then for any $t > 0$,*

$$\Pr\left\{\left|\sum_{i=1}^{n} X_i - np\right| \geq t\right\} \leq 2e^{-\frac{2t^2}{n}} \tag{2.9}$$

Proof. Let $X = \sum_{i=1}^{n} X_i$. Formula (2.9) can be derived from the observation that for any $s > 0$,

$$\begin{aligned} \Pr\{X \geq np + t\} &= \mathcal{E}\left[I_{\{X-(np+a)\geq 0\}}\right] \leq \mathcal{E}\left[e^{s(X-(np+t))}\right] \\ &= e^{-s(np+t)}\mathcal{E}\left[e^{sX}\right] = e^{-s(np+t)}(pe^s + (1-p))^n. \end{aligned}$$

□

The Chernoff bound has been generalized to the cases where $\{X_i, 1 \leq i \leq n\}$ are (not necessarily Bernoulli) random variables and the sum $\sum_{i=1}^{n} X_i$ is replaced by a function f of the variables that satisfies the so-called "bounded differences" condition.

Lemma 2.1.5 (McDiarmid [110]). *Let $\{X_1, \ldots, X_n\}$ be a sequence of independent random variables with each X_i defined on a probability space $(\Omega_i, \mathcal{A}, \Pr_i)$. Let $f : \prod_{i=1}^{n} \Omega_i \to R$ be a function such that*

$$|f(\omega) - f(\omega')| \leq c_i$$

whenever $\omega, \omega' \in \prod_{i=1}^{n} \Omega_i$ differ only in the i-th coordinate. Then for any $t > 0$,

$$\Pr\{|f(X_1, \ldots, X_n) - \mathcal{E}[f(X_1, \ldots, X_n)]| > t\} \leq 2e^{-\frac{2t^2}{\sum_i c_i^2}}. \qquad (2.10)$$

The Chernoff bound can also be generalized to the cases where the sequence of random variables $\{X_i, 1 \leq i \leq n\}$ are dependent. The most famous result is Hoeffding-Azuma's inequality for martingale-differences sequences [14, 131]. It is worth noting that the exponent in McDiarmid's bound in (2.10) is better by a factor of 4 than those obtained from Hoeffding-Azuma's inequality [30]. See [91, 135] for more recent progress on dependent variables and functions that violate the "bounded differences" condition.

2.2 Theory of Random Graphs

The theory of random graphs, founded by Erdös and Rényi [59], is at the core of the probabilistic method. It deals with various structural graph properties in random models of graphs. Popular random models of graphs include the original Erdös-Rényi random graph [30, 120], the random regular graph [100, 139], and the more recent scale-free random graph [13, 56]. In the following, the term "random graph" will always refer to Erdös-Rényi random graph model.

Definition 2.2.1 (Models of Random Graphs). *Let V be a set of vertices with $|V| = n$.*

1. ***Constant-Probability Model*** $G(n,p)$. *In this model, each of the* $\binom{n}{2}$ *potential edges appears in the graph independently with probability* p.

2. ***Uniform Model*** $G(n,m)$. *In this model, the graph contains a set of* m *edges selected uniformly at random without replacement.*

It is not hard to see that for a given graph $G = G(V,E)$ with $|E| = m$,

$$\Pr\{G(n,p) = G\} = p^m(1-p)^{\binom{n}{2}-m}$$

and

$$\Pr\{G(n,m) = G\} = \binom{\binom{n}{2}}{m}^{-1}.$$

A *graph property* is a subset of graphs. A graph property $\mathcal{Q}$ is said to be *monotone increasing* if for any two graphs G and H such that if $G \in \mathcal{Q}$ and $G \subset H$, we have $H \in \mathcal{Q}$. For a monotone increasing graph property $\mathcal{Q}$, the following result is straightforward:

$$\Pr\{G(n,m_1) \in \mathcal{Q}\} \leq \Pr\{G(n,m_2) \in \mathcal{Q}\} \text{ if } m_1 < m_2. \tag{2.11}$$

To see this, let $\mathcal{A}_1 \subset \mathcal{Q}$ be the set of graphs with m_1 edges and $\mathcal{A}_2 \subset \mathcal{Q}$ be the set of graphs with m_2 edges. Each graph in $\mathcal{A}_1$ corresponds to $\binom{\binom{n}{2}-m_1}{m_2-m_1}$ graphs in $\mathcal{A}_2$, while each graph in $\mathcal{A}_2$ corresponds to at most $\binom{m_2}{m_1}$ graphs in $\mathcal{A}_1$. Thus, $|\mathcal{A}_1|\binom{\binom{n}{2}-m_1}{m_2-m_1} \leq |\mathcal{A}_2|\binom{m_2}{m_1}$.

For many problems, it is much easier to work with the constant-probability model than with the uniform model. Fortunately for monotone increasing properties and under very mild conditions on p and m, the two models are probabilistically equivalent. See [30, 120] for a detailed discussion.

In the study of (constraint) satisfiability problems, hypergraphs are also widely used. Random models of hypergraphs can be defined in a similar way.

Definition 2.2.2 (Random Hypergraphs).

1. *A hypergraph* $\mathcal{G} = \mathcal{G}(V,E)$ *is a pair* (V,E) *where* V *is the set of vertices and* E *is a collection of subsets of* V*, called hyperedges. A hypergraph is* k*-homogenous*[1] *if its hyperedges are all of cardinality* k.

2. ***Constant-Probability Model*** $\mathcal{G}^k(n,p)$. *In this model, each of the* $\binom{n}{k}$ *potential edges appears in the graph independently with probability* p.

3. ***Uniform Model*** $\mathcal{G}^k(n,m)$. $\mathcal{G}^k(n,m)$ *is a random hypergraph consisting of* m *hyperedges chosen uniformly at random without replacement from the collection of all the* $\binom{n}{k}$ *potential hyperedges.*

[1] In the literature, the corresponding concept is called k-uniform.

Note that the random graph $G(n,m)$ is just the random 2-homogenous hypergraph $\mathcal{G}^2(n,m)$.

A main theme in the study of random graphs is the threshold phenomenon and phase transitions that characterize the abrupt change of monotone graph properties when some parameter of the random model crosses a critical value (or function). A detailed account can be found in [30, 120] and the references therein.

One of the most interesting results on the phase transitions of random graphs is the abrupt change of the component structure of the random graph $G(n,m)$ at the threshold $\frac{m}{n} = \frac{1}{2}$. It is well-known that for $\frac{m}{n} < \frac{1}{2}$, $G(n,m)$ consists of small-sized tree and unicyclic components **whp** , while for $\frac{m}{n} > \frac{1}{2}$, $G(n,m)$ has a "giant" component of size $\Theta(n)$ **whp** [30, 59, 120]. A similar result also holds for the random k-homogenous hypergraph $\mathcal{G}^k(n,m)$.

The *excess* of a k-homogenous hypergraph $\mathcal{G} = \mathcal{G}(V,E)$ is defined as

$$ex(\mathcal{G}) = (k-1)|E| - |V|. \tag{2.12}$$

A connected hypergraph $\mathcal{G}$ is called a *hypertree* if $ex(\mathcal{G}) = -1$ and a *unicycle* if $ex(\mathcal{G}) = 0$.

Lemma 2.2.1 ([97]). *If $\frac{m}{n} < \frac{1}{k(k-1)}$, then* **whp** *the random k-homogenous hypergraph $\mathcal{G}^k(n,m)$ consists of only hypertrees and unicyclic components.*

2.3 Decision Problems, Computational Complexity, and Proof Complexity

A *decision problem* consists of a pair $(\Xi, \mathcal{L})$ where Ξ is a set of *problem instances* and $\mathcal{L}$, called a *property* or a *language*, is a subset of Ξ. The question is to decide the membership of a given problem instance in the language $\mathcal{L}$. The set of problem instances of size n is denoted by Ξ^n. A decision problem $(\Xi, \mathcal{L})$ together with a partial order $\prec$ on Ξ is said to be monotone (with respect to $\prec$) if one of the following conditions is satisfied:

1. $\forall I_1, I_2 \in \Xi$, if $I_1 \prec I_2$ and $I_1 \in \mathcal{L}$, then $I_2 \in \mathcal{L}$; or
2. $\forall I_1, I_2 \in \Xi$, if $I_1 \prec I_2$ and $I_2 \in \mathcal{L}$, then $I_1 \in \mathcal{L}$.

The graph coloring problem and the Hamiltonian cycle problem are famous examples of decision problems that are monotone with respect to the partial order defined by the inclusion of edge sets, while SAT and CSP are monotone decision problems with respect to the partial order defined by the inclusion relation of subsets of clauses or constraints.

A *witness*, or a *proof*, for a problem instance is a piece of properly encoded information concerning the membership of the instance. A decision problem

$(\Xi, \mathcal{L})$, or more precisely the language $\mathcal{L}$, is in NP if there is an polynomial-time procedure A such that

$$\mathcal{L} = \{I \in \Xi : \text{ there exists a witness } y \text{ such that } A \text{ accepts } (I, y)\}.$$

The notion of *NP-completeness* was first formulated by S. Cook and L. Levin in early 1970s, and has since then played a very important role in the study of computational complexity and in the analysis of algorithms. A decision problem is *NP-complete* if it is in NP and any decision problem in NP can be reduced to it polynomially.

A central task in the study of computational complexity is to provide a classification of various decision problems in terms of the required computing resources. The subclass P of the NP decision problems is a class of problems that can be solved in polynomial time. Understanding the relation between the classes P and NP is one of the main driving forces in the theory of computational complexity.

The theory of proof complexity, on the other hand, deals with the size of the proofs or witnesses of a decision problem. The class co-NP is the set of decision problems that are the complement of some NP decision problem. A *proof system* for a language $(\Xi, \mathcal{L})$ in co-NP is a polynomial-time computable procedure $S(x, \pi) : \Xi \times \Sigma^* \to \{0, 1\}$ satisfying the following properties

1. Soundness: $\exists \pi \in \Sigma^*$ such that $S(x, \pi) = 1 \Rightarrow x \in \mathcal{L}$;
2. Completeness: $x \in \mathcal{L} \Rightarrow \exists \pi$ such that $S(x, \pi) = 1$.

We call $\pi \in \Sigma^*$ a proof and denote its size by $|\pi|$. The central question in the theory of proof complexity is that of "NP versus co-NP", i.e., "Does every co-NP problem have a polynomial size proof?" A line of research aimed at resolving the NP versus co-NP problem is to establish lower bounds on the proof size for proof systems of increasing strength.

2.3.1 Resolution Proof System

Proof systems for the propositional satisfiability problem (SAT) in propositional logic are of special theoretical and practical interest.

Definition 2.3.1 (Literals, Clauses, and Formulas). *In propositional logic, variables take two possible values 0 (false) and 1 (true). A literal is either a variable x or its negation $\overline{x}$.*

1. *A clause is a disjunction of a set of literals. A clause that contains k (distinct) literals is called a k-clause.*
2. *A conjunctive normal form (CNF) formula is the conjunction of a set of clauses. A CNF formula that contains only k-clauses is called a k-CNF formula. It is also common and convenient to simply regard a CNF formula as a collection of clauses.*

The *resolution proof system* is a special proof system for the language of unsatisfiable CNF formulas, co-CNF. In this system, a *resolution proof* π consists of a sequence of clauses $\{C_1, \cdots, C_s\}$ where the last clause C_s is empty (i.e., a contradiction) and each $C_i, 1 \leq i \leq s$, is either a clause from the original formula or a clause derived from two precedent clauses by the following derivation rule:

1. Resolution Rule: Derive $C \vee D$ from a pair of clauses $\{C \vee x, D \vee \overline{x}\}$ where x is a literal.

2. Weakening Rule: Derive $C \vee D$ from C for any pair of clauses $\{C, D\}$ [2].

A resolution proof can be represented as a directed acyclic graph (DAG) where the vertices are the clauses in the proof and each vertex has two in-edges from the two premise clauses. A proof is said to be *tree-like* if its associated DAG is a tree.

Definition 2.3.2. *The size $|\pi|$ of a resolution proof π is the number of clauses in it. The resolution complexity* $\mathrm{RES}(\mathcal{F})$ *of a CNF formula $\mathcal{F}$ is the minimum size of a resolution proof of $\mathcal{F}$,*

$$\mathrm{RES}(\mathcal{F}) = \min\{|\pi| : \pi \text{ is a resolution proof }\}$$

The minimum size of a tree-like resolution proof of a formula $\mathcal{F}$ is denoted by $\mathrm{RES}_T(\mathcal{F})$.

2.3.2 Hard Formulas for Resolution

The study of the resolution proof system has a long history. Over the past 30 years, there has been much effort in constructing hard examples of CNF formulas that have exponential resolution complexity. Earlier work includes lower bounds on specially-constructed CNF formulas such as the Tseitin formulas, the pigeonhole principle, and the clique principle [24, 31, 26, 83].

Example 2.3.1 (The Pigeonhole Principle). *The pigeonhole principle states that it is impossible to put $n+1$ pigeons into n holes so that each pigeon is in a distinct hole. The pigeonhole principle can be encoded as an unsatisfiable CNF formula as follows:*

$$\mathbf{PHP_n} = \left\{ \begin{array}{ll} x_{i1} \vee \cdots \vee x_{in}, & \text{for } 1 \leq i \leq n+1 \\ \overline{x}_{ik} \vee \overline{x}_{jk}, & \text{for } 1 \leq i, j \leq n+1 \text{ and } 1 \leq k \leq n. \end{array} \right\} \quad (2.13)$$

where for pair of (i, j), $x_{ij} = 1$ means that the pigeon i is in the hole j. Haken [83] established an exponential lower bound on the resolution complexity of $\mathbf{PHP_n}$, *i.e.,*

$$\mathrm{RES}(\mathbf{PHP_n}) = 2^{\Omega(n)}.$$

[2] The weakening rule is only for the purpose of convenience and is not essential.

This result has been later generalized to the case where the number of pigeons is any number larger than the number of holes (see, e.g., [26]).

Example 2.3.2 (Tseitin formula). *The Tseitin formula encodes the basic fact in graph theory that the sum of the vertex degrees of a graph cannot be odd. Let $G = G(V, E)$ be a graph and $\sigma : V(G) \to \{0, 1\}$ a weight defined on each vertex. To construct the Tseitin formula, we associate a variable x_e with each edge $e \in E$ and define for each vertex v the following boolean expression*

$$PARITY_v : \bigotimes_{e \in E : v \in e} x_e \equiv \sigma(v) (mod 2).$$

Let T_v be the set of clauses that is equivalent to $PARITY_v$. The Tseitin formula $T(G, \sigma)$ is defined to be the set of clauses

$$T(G, \sigma) = \bigcup_v T_v.$$

Urquhart [133] proved that

1. *$\sum_v \sigma(v)$ is odd $\Rightarrow T(G, \sigma)$ is unsatisfiable; and*
2. *If G is connected, then $T(G, \sigma)$ is unsatisfiable $\Rightarrow \sum_v \sigma(v)$ is odd.*

The resolution complexity $\mathrm{RES}(T(G, \sigma))$ has an exponential lower bound if the connected graph G satisfies some expanding property [133, 26].

Example 2.3.3 (The Clique Principle). *It is obvious that a k-clique cannot be subgraph-isomorphic to any graph that is $(k-1)$ colorable. The clique principle states the even more obvious fact that it is impossible for a k-clique to be subgraph-isomorphic to a maximally $(k-1)$-colorable graph. It has been proved that the CNF encoding of the clique principle requires an exponential size of small-weight Cutting Planes proof, a restricted Cutting Planes proof that still includes the resolution proof as its special case [31].*

Another source of hard instances for resolution is randomly-generated CNF formulas. Let $\mathcal{F}^k_{n,m}$ be a k-CNF formula on n variables consisting of m randomly generated k-clauses. In their seminal paper, Chvátal and Szemerédi [37] proved that for any fixed $\frac{m}{n} = c$ and $k \geq 3$, there is a constant $\kappa > 0$ such that

$$\lim_n \Pr \left\{ \mathrm{RES}(\mathcal{F}^k_{n,m}) \geq 2^{\kappa n} \right\} = 1.$$

Recently, upper bounds as well as improved lower bounds with an explicit estimation of the dependency of κ on the ratio $\frac{m}{n}$ were established in [22, 26]. In particular, the improved lower bounds imply that the typical resolution complexity of $\mathcal{F}^k_{n,m}$ is still super-polynomial for some m that increases as a polynomial function of n:

Theorem 2.3.1 ([22, 26]). *For $k \geq 3$ and $\epsilon > 0$, there is a constant $\gamma > 0$ such that*

$$\lim_n \Pr\left\{ \mathrm{RES}(\mathcal{F}^k_{m,n}) \geq 2^{n^\gamma} \right\} = 1, \text{ if } m \leq n^{\frac{k+2}{4}-\epsilon}.$$

This is in contrast to the results that $\mathrm{RES}(\mathcal{F}^k_{m,n})$ is polynomial if $m = \Omega(n^{k-1})$ [60].

In [4], the resolution complexity of a class of mixed random CNF formulas $\mathcal{F}^\epsilon_{n,m}$ is studied that contains $m = cn$ random 3-clauses and $(1+\epsilon)n$ 2-clauses. It was shown that as long as $\epsilon < 0$, i.e., there are less than n 2-clauses, the random CNF formula $\mathcal{F}^\epsilon_{n,m}$ has an exponential resolution complexity **whp** . On the other hand, for $\epsilon > 0$, $\mathcal{F}^\epsilon_{n,m}$ has a polynomial resolution complexity **whp** since the set of $(1+\epsilon)n$ 2-clauses alone makes the formula unsatisfiable. Also in [4] are some lower bounds on the running time of backtracking search algorithms for satisfiable random CNF formulas. These lower bounds are based on the observation that backtracking algorithms on random CNF formulas will create certain types of mixed random CNF formulas during their execution.

Most of the work on establishing exponential lower bounds exploits the relation between the minimum proof size and the minimum of the maximum clause length of all the resolution proofs. The idea has been formalized by E. Ben-Sasson [27, 26] as the so-called *width-method* which we briefly discuss below.

Definition 2.3.3 ([27, 26]). *1. A clause mentions a variable x if it contains either x or its negation $\overline{x}$. The length $|C|$ of a clause C is the number of variables that C mentions. The width of a set of clauses is the maximum length of a clause in the set. In particular, we use $w(\mathcal{F})$ to denote the width of the CNF formula $\mathcal{F}$.*

2. The width of deriving a clause C from a CNF formula $\mathcal{F}$, denoted as $w(\mathcal{F} \vdash C)$, is the minimum width of all the possible derivations.

3. The width of deriving the empty clause $w(\mathcal{F} \vdash \emptyset)$ is called the refutation width of $\mathcal{F}$.

Theorem 2.3.2 ([27, 26]). *For any CNF formula $\mathcal{F}$, we have*

$$\mathrm{Res}(\mathcal{F}) = e^{\Omega(\frac{(w(\mathcal{F}\vdash\emptyset)-w(\mathcal{F}))^2}{n})}. \tag{2.14}$$

and

$$\mathrm{Res}_T(\mathcal{F}) = 2^{(w(\mathcal{F}\vdash\emptyset)-w(\mathcal{F}))}. \tag{2.15}$$

Theorem 2.3.2 reduces the task of establishing exponential lower bounds to that of proving the existence of a clause of width linear in n in any refutation proof. For many random decision problems, this latter task amounts to showing that (1) the minimum unsatisfiable subproblem has a "large size" **whp** , and (2) satisfiable subproblems with a "medium size" are typically "sparse" and thus have a high degree of local consistency, resulting in some long clauses in any resolution proof.

2.3.3 DPLL Procedure and Resolution Complexity

The *Davis-Putnam procedure (DP procedure)* proposed by Davis and Putnam [49] is a dynamic-programming algorithm to generate a special type of resolution proof. Given a CNF formula and an ordering of the variables, the DP procedure eliminates the current variable by performing all the possible resolutions on the variable, adding all the resolvents to the formula, and removing all the clauses that mention the current variable, until the empty clause is produced or no more resolvents can be formed. The basic DP-procedure is described in Algorithm 2.1.

Algorithm 2.1 Davis-Putnam procedure (DP procedure)

Input: A CNF formula $\mathcal{F}$ on $\{x_1, \cdots, x_n\}$

for $(i = 1; i \leq n; i^{++})$ **do**
  Resolve each pair of clauses of the form $C \vee x_i$ and $D \vee \overline{x}_i$ in $\mathcal{F}$, and add the resolvent $C \cup D$ to $\mathcal{F}$;
  Remove all the clauses that mention x_i;
  if ($\mathcal{F}$ contains the empty clause) **then**
    return UNSATISFIABLE;
  end if
end for
return SATISFIABLE;

The DP procedure is sound and complete, but its time and space complexities is exponential in the "tree-width" of the underlying graph structure of the CNF formula under consideration [53].

Theorem 2.3.3. *DP procedure is sound and complete.*

Proof. The soundness follows directly from the soundness of the resolution rule. To see the completeness, let $\mathcal{F}$ be the original formula and $\mathcal{F}_{i+1}$ be the CNF formula after Step i of the DP procedure where variable x_i is eliminated. We will show that if the DP procedure (Algorithm 2.1) does not create an empty clause, then $\mathcal{F}$ is satisfiable.

Assume that the DP procedure returns SATISFIABLE after the variable $x_m, 1 \leq m \leq n-1$ has been eliminated. Then, all the literals in $\mathcal{F}_{m+1}$ must be pure, i.e., each variable mentioned in $\mathcal{F}_{m+1}$ appears as either a positive literal or a negative literal in all the clauses of $\mathcal{F}_{m+1}$. Thus, $\mathcal{F}_{m+1}$ is satisfiable. Since $\mathcal{F}_{i+1}$ contains all the resolvents $C \vee D$ where $C \cup \{x_i\}$ and $D \cup \{\overline{x}_i\}$ are two clauses in $\mathcal{F}_i$, we know that any assignment $A(i+1)$ to the variables $\{x_k, k \geq i+1\}$ that satisfying $\mathcal{F}_{i+1}$ can be extended to an assignment $A(i)$ to the variables $\{x_k, k \geq i\}$ to satisfy $\mathcal{F}_i$. Otherwise, there must be a pair of clauses $C \vee \{x_i\}, D \vee \{\overline{x}_i\}$ such that $A(i+1)$ does not satisfies $C \vee D$. By induction on i, $\mathcal{F}$ is satisfiable. □

Closely related to the DP procedure is the Davis-Putnam-Logemann-Loveland (DPLL) algorithm implemented in [48]. DPLL has been the basis of most of the modern high-performance solvers for the propositional satisfiability problem.

Although being a typical backtracking search algorithm, DPLL can be viewed as an algorithm that constructs a refutation proof of the given CNF formula. The execution of DPLL can be represented as a rooted binary tree in which each internal node corresponds to a recursion call and is labelled by the branch variable. The two out-edges of an internal node correspond to the two possible assignments (0 or 1) to the branch variable. Each path from the root to a leaf defines a (partial) assignment to the variables. For an unsatisfiable CNF formula, every leaf is a "failure leaf", i.e., at least one of the clauses in the formula is falsified by the corresponding assignment. A refutation proof can be constructed from the execution of the DPLL algorithm as follows. First, we label each leaf with a clause falsified by the corresponding assignment. Then, recursively we label each node with the resolvent of the two clauses that label the two children of the node. The root will be labelled by the empty clause.

The above discussion implies that the time complexity of the DPLL algorithm is lower bounded by the (tree-like) resolution complexity of the CNF formula. Nonetheless, there has been much effort to improve the efficiency of the DPLL-like algorithm by using more clever data structures, devising better heuristics, and incorporating more powerful reasoning mechanisms. The payoff of these efforts can be observed from the yearly SAT-solver competition and success stories of various industrial applications [82].

The basic structure of modern DPLL-based SAT-solvers is presented in Algorithm 2.2. The performance of a specific solver depends on the implementation of the three procedures UNIT-PROPAGATION, CONFLICT-ANALYSIS, and BRANCHING.

UNIT-PROPAGATION prunes the search space by propagating the consequences of assigning a truth-value to a variable x. It employs a basic form of look-ahead strategy similar to that of maintaining arc-consistency in constraint satisfaction solving techniques. First, clauses that mention x are processed to record the effect of such an assignment. Then, *unit clauses*—clauses that under the current partial assignment have no satisfied literal but exactly one unvalued literal—are detected and relevant unassigned variables are assigned to make these clauses satisfied. This step is repeated until a clause becomes falsified or there are no more unit clauses under the current partial assignment. It turns out that UNIT-PROPAGATION, though a powerful mechanism for pruning the search space, accounts for a large fraction of the overall running time of a SAT solver [55]. This is mainly because UNIT-PROPAGATION needs to maintain the status of all the clauses after each variable assignment. Several clever data structures have been proposed to speed up UNIT-PROPAGATION, and the most effective one is the so-called *watched-literals* [142].

CONFLICT-ANALYSIS, also known as clause-learning or nogood-learning, is invoked when the current partial assignment results in a contradiction and the algorithm needs to backtrack [18, 20, 141]. By analyzing the implication relationships among the variable assignments, represented as an *implication graph* [141], new clauses that explain the cause of the current failure are added to the original CNF formula. These new clauses, though redundant in terms of the satisfiability of the original formula, prune a subspace that might otherwise be searched repeatedly by the algorithm. It can be shown that algorithms with clause-learning mechanism are exponentially stronger than tree-like resolution algorithms such as DPLL [23]. Of course, from a more practical perspective, the time and memory overhead of clause-learning also have a significant impact on the overall performance of a solver.

In BRANCHING, also known as splitting, various heuristics can be used to select the next variable to assign a truth value. A variable selected in BRANCHING is called a *branch variable* or *decision variable* while variables that are assigned values as a result of UNIT-PROPAGATION are called *implied variables*. Over the years, many heuristics have been proposed and most of them have been summarized in [82, 55].

Algorithm 2.2 DPLL Algorithm: **DPLL**$(\mathcal{F}, A)$

// Input: $\mathcal{F}$, a formula on $\{x_1, \cdots, x_n\}$; A, a list of partial assignment.

```
A ← UNIT-PROPAGATION;
if A falsifies a clause then
  CONFLICT-ANALYSIS;
  return FAILURE;
else if all the clauses are satisfied then
  return SUCCESS;
end if
x ← BRANCHING;
if DPLL(F, A ∪ {x = 1}) = SUCCESS then
  return SUCCESS;
else
  return DPLL(F, A ∪ {x = 0});
end if
```

2.4 Phase Transitions and Typical-case Complexity

The significance of the notion of NP-completeness lies in the fact that unless P = NP, no polynomial time algorithm exists for NP-complete problems. Thus, a proof of NP-completeness of a decision problem is a strong evidence that

the problem is hard. There are several approaches to NP-complete problems. One may try (1) to identify subclasses of the problem for which polynomial algorithms exist; (2) to design efficient heuristics to solve the problems; (3) to develop approximation algorithms with a performance guarantee for NP-hard problems; and (4) to understand the typical-case/average-case complexity of the problem instances under some probability distribution. The study of phase transitions and typical-case complexity belongs to the fourth approach, but is also related to the second approach.

A random model of a decision problem $(\Xi, \mathcal{L})$ is a triple $(\Xi, \mathcal{L}, \mathcal{P}_\Xi)$ where $\mathcal{P}_\Xi$ is a probability measure on Ξ. When the decision problem is clear from the context, we will simply call the probability measure $\mathcal{P}_\Xi$ a *random decision problem.*

Studying properties of random problems and designing efficient (randomized) algorithms to solve hard problems **whp** (or on average) have long been a topic of great interest in discrete mathematics and theoretical computer science. See, for example, [37, 61, 66, 68, 104, 107, 131] and the references therein. While being elegant and interesting in its own right, the study did not attract wide attention from the public of computer science in general and AI community in particular, until the work of Cheeseman et al. [35] in early 1990's that pointed out an explicit connection between the pattern of typical-case complexity and the phase transition of the solution probability of randomly-generated instances of NP-complete problems. Over the past decade, we have gained much insight into both the nature of the phase transition of NP-complete problems and the question of where the really hard problems are and why they are hard. Computer scientists, mathematicians, and theoretical physicists all have contributed to this progress. In the rest of this section, we give a brief overview of the study of the phase transitions in several NP-complete problems, including SAT, Graph Coloring, Hamiltonian Cycle, and Number Partitioning (See, Problem 2.1). For each of the four problems, the specific topics and results selected to discuss is based on my own interest and is perhaps subjective.

2.4.1 SAT

SAT is one of the most popular and important NP-complete problems in computational complexity and AI [82]. It is the first problem that was shown to be NP-complete and has been the focus of the study of phase transitions and typical-case complexity of NP-complete problems.

An instance of SAT is a CNF formula and the question is to decide whether there is a truth assignment that satisfies the formula. When the instances are restricted to k-CNF formulas, we call the problem k-SAT. It is well-known that k-SAT is NP-complete for $k \geq 3$ and can be solved in linear time for $k = 2$. One of the well-studied random models for k-SAT is $\mathcal{F}_{n,m}^k$ on n variables which consists of m clauses selected uniformly at random from the set of all $2^k\binom{n}{k}$

Problem 2.1 NP-complete Problems

Satisfiability (SAT)
INSTANCE: $F \in \Xi = \{$ All the CNF formulas$\}$
QUESTION: Is $F \in L = \{$ All the satisfiable formulas$\}$?

Graph k-Colorability
INSTANCE: A graph $G(V, E)$ and an integer $k \geq 1$.
QUESTION: Is there a coloring $c : V \rightarrow \{1, 2, \cdots, k\}$ such that $c(v_1) \neq c(v_2)$ if $(v_1, v_2) \in E$?

Hamiltonian Cycle
INSTANCE: A graph $G(V, E)$.
QUESTION: Does G have a Hamiltonian cycle?

Number Partition
INSTANCE: A set of integers I.
QUESTION: Is there a partition I_1, I_2 of I such that $\sum_{i \in I_1} i = \sum_{i \in I_2} i$?

possible k-clauses. The ratio $r = \frac{m}{n}$ is called the *clause-variable ratio* or the *clause density* of $\mathcal{F}^k_{n,m}$.

Intuitively, formulas with a large clause-variable ratio are hard to be satisfied, while formulas with a small clause-variable ratio could have many satisfying solutions. Experimental studies in [114, 101] indicated that at $r \approx 4.2$ there is an abrupt change of the probability that $\mathcal{F}^3_{n,m}$ has a solution—the probability is asymptotically one for $r < 4.2$ and zero for $r > 4.2$. This leads to the following famous conjecture:

SAT Threshold Conjecture: There is a constant r_k, called the *satisfiability threshold*, such that

$$\lim_n \Pr\{\, \mathcal{F}^k_{n,m} \text{ is satisfiable } \} = \begin{cases} 1, & \text{if } r < r_k \\ 0, & \text{if } r > r_k \end{cases}$$

After more than ten years of work, the above conjecture is far from being settled. However, much insight into the hardness pattern and its algorithmic impact has been gained from the effort to try to understand the behavior around the threshold and to improve the upper and lower bounds on the threshold.

Upper and Lower Bounds

Upper bounds on the SAT threshold can be established by Markov's inequality. For example, $\forall a \in \{0,1\}^n$ let I_a denote the indicator function of the event that the assignment a satisfies $\mathcal{F}^3_{n,m}$, and let $I = \sum_a I_a$. We have

$$\Pr\{I > 0\} \leq \mathcal{E}[I] = \sum_a \mathcal{E}[I_a] = 2^n(\frac{7}{8})^m.$$

Thus, $\mathcal{F}^3_{n,m}$ is unsatisfiable **whp** if $m/n > \log_{8/7} 2 = 5.191$. This gives an upper bound for the SAT threshold. By taking into consideration the intrinsic structure of the solution space, better upper bounds can be obtained. See [57] for an account on a series of hard work that improves the upper bound from 5.191 to 4.596.

Lower bounds on the threshold are usually obtained by analyzing polynomial-time algorithms based on the unit clause heuristic: (1) If there are any clauses containing only one literal, then pick one of them and satisfy it. Otherwise, randomly pick an unset variable and assign it to TRUE (or FALSE) randomly and uniformly. By analyzing conditions under which these algorithms succeed, lower bounds can be established [3].

Lower bounds can also be established by the second moment method. The difficulty in using the second moment method lies in the fact that the random variable under consideration is usually a sum of a set of random variables that are only "close" to being independent. To bound the variance of such a random variable, combinatorial structures intrinsic to the problem should be utilized in a smart way [12, 14].

Sharpness of the Phase Transition

In addition to the location of the critical point of the phase transition, the sharpness of the phase transition is also interesting. Roughly speaking, a phase transition for a combinatorial property is sharp if the length of the transition interval tends to zero faster than the critical parameter itself. A sufficient condition for a combinatorial property to have a sharp phase transition has been established [65]. The condition basically indicates that in order to have a sharp transition, there should not exist small signatures (properties that can be determined locally) that can probabilistically approximate the property under consideration.

Backbones and Complexity

In the statistical mechanics approach to the random SAT phase transition, a Boolean variable is identified with a binary variable, called a spin, that takes its values on $\{-1, 1\}$ (-1 for FALSE and 1 for TRUE). A CNF formula $\mathcal{F}$

is associated with an energy function $\mathbf{E}[\mathcal{F}, S], S \in \{-1, 1\}^n$, defined on the possible assignments to the binary variables, indicating the number of clauses not satisfied by the assignment.

To investigate the behavior of the optimum of the energy function and the structure of the space of the optimal solutions, statistical physicists treat the SAT problem as a system of spins whose configuration is governed by the Boltzmann distribution

$$p(S) = \frac{1}{Z} e^{-\frac{1}{T}\mathbf{E}[\mathcal{F}, S]}$$

and its low temperature limit as T tends to zero. Note that this distribution is just a vehicle to carry out statistical mechanics analysis and has nothing to do with the randomness in the random SAT formula, which is called the *quenched disorder* in physics.

Analytical techniques from statistical mechanics can be used to analyze the deep relations among the minimum of the energy function $\mathbf{E}[\mathcal{F}, S]$, the Boltzmann distribution of the SAT system, and the probability distribution of the random SAT. These analyses have revealed interesting structural properties of the space of the optimal solutions and help explain why problem instances are hard at phase transitions. Most notably is the notion of *backbone variables* [109, 117].

For each variable x_i, let m_i be the average value of the corresponding spin over all the optimal assignments. Note that $|m_i| = 1$ implies that the variable x_i is fully constrained, i.e., it has to be assigned to the same value in every optimal solution. In this case, the variable is called a backbone variable or a frozen variable [44, 109].

For a random SAT, m_i is a random variable in $[-1, 1]$. Statistical mechanics analysis shows that the asymptotic behavior of the fraction of backbone variables is quite different at 2-SAT phase transition and 3-SAT phase transition. For random 2-SAT, it changes smoothly across the threshold, while for random 3-SAT, the fraction of backbone variables jumps discontinuously from zero to a positive constant at the phase transition. That is to say, right above the clause density threshold, a constant fraction of the variables suddenly become fully constrained. There is also theoretical and empirical evidence showing that a close relation exists between the behavior of the backbone and backtracking-style search algorithms as well as random local search algorithms. See, for example, the work on the behavior of backbones in the $2 + p$-SAT problem where an instance of the problem consists of a mixture of 2-CNF clauses and 3-CNF clauses [117].

Analysis also reveals interesting characteristics about the structure of the space of the optimal solutions in the satisfiable region

1. When the clause density is well below the phase transition threshold, the optimal solutions form a single cluster and these solutions are all characterized by a common distribution;

2. When the clause density is close to the phase transition threshold, the single cluster of optimal solutions break up into exponentially many smaller clusters. While the distances between solutions in different clusters remains constant, solutions in a single cluster become more and more similar to each other as the clause density increases.

2.4.2 Graph Coloring

The study of the phase transition of the graph coloring problem is based on the standard random graphs $G(n,p)$ or $G(n,m)$ as defined in Section 2.2. Similar to random SAT, the phase transition of k-colorability is sharp [5]. Currently known upper and lower bounds on the threshold are summarized in the following theorem [3]:

Theorem 2.4.1 ([2, 10, 124]). *(1) For the 3-colorability problem,*

$$\lim_n \Pr\{\, G(n, p = c/n) \text{ is 3-colorable} \,\} = \begin{cases} 1, & \text{if } c < 4.03 \\ 0, & \text{if } c > 5.05 \end{cases}$$

(2) For any $k > 3$,

$$\lim_n \Pr\{\, G(n, p = c/n) \text{ is } k\text{-colorable} \,\} = \begin{cases} 1, & \text{if } c \leq k \\ 0, & \text{if } c > 2k \ln k \end{cases}$$

Algorithm 2.3 Greedy k-Coloring

Input: A graph $G(V, E)$ and a set of available colors represented as integers.

Find a vertex order $\{v_1, v_2, \cdots, v_n\}$.
for $(i = 1; i \leq n; i^{++})$ **do**
  Assign to v_i the smallest color that is consistent with the colors already assigned to the vertices $v_1, \cdots, v_{i-1}$.
end for

The k-colorability can be decided in constant expected time for random graphs with a constant edge probability [104], largely because of the appearance of $(k+1)$-cliques. As the edge probability becomes smaller, the hardness of the k-colorability problem increases significantly.

In the analysis of the typical behavior of the random k-colorability problem below the colorability threshold, variants of the greedy coloring algorithm (Algorithm 2.3) have been widely used. Given a vertex ordering, the greedy coloring algorithm iteratively assigns to the next vertex the first available color. Different vertex orderings give rise to different heuristics which are used to analyze the typical behavior of the k-colorability problem on random graphs. Let $G(n, \frac{c}{n})$ be the random graph with the edge probability $p = \frac{c}{n}$, we have the following cases.

[3] See [11] for the more recent progress

1. **(Arbitrary Vertex Ordering)** Algorithm 2.3 with an arbitrary vertex ordering can find a k-coloring ($k \geq 3$) **whp** if $c < 1$. This is because the random graph with $c < 1$ consists of only trees and unicycles **whp** so that the number of the colors used by Algorithm 2.3 never exceeds 3. Algorithm 2.3 has also been shown to be effective on dense random graphs [104].

2. **(k-Core Heuristic)** Earlier lower bounds on the k-colorability threshold were obtained by analyzing the threshold of the existence of a k-core in random graphs. A k-core is defined to be the unique maximal induced subgraph with minimum vertex degree at least k [124]. The k-core heuristic determines a vertex ordering $\{v_1, v_2, \cdots, v_n\}$ such that for any i, the vertex degree of v_i in the subgraph induced on the vertex set $V \setminus \{x_{i+1}, \cdots, v_n\}$ is less than k. If a graph does not contain a k-core, such an ordering exists and can be found in polynomial time. It follows that greedy coloring with the k-core heuristic finds a coloring **whp** in polynomial time for any c below the threshold of the appearance of the k-core. In the case of $k = 3$, the k-core threshold is approximately 3.35 [124].

3. **(Brelaz's Heuristic)** In Brelaz's heuristic, the vertex ordering is constructed dynamically. At each step, a vertex with the most distinctly colored neighbors is selected and is assigned the smallest available color. Variants of Greedy k-Coloring with Brelaz's heuristic has been analyzed in [2, 10], resulting in the best known lower bound $c > 4.03$ on the k-colorability threshold of random graphs.

When the edge probability is close to the colorability threshold, the k-colorability problem becomes exponentially hard. The typical-case behavior of backtracking algorithms has been extensively studied. Bender and Wilf proved that the running time of a simple backtracking algorithm is $2^{\Theta(1/p)}$. In [21, 112], exponential lower bounds are established for the resolution complexity of the k-colorability problem, indicating that most backtracking graph coloring algorithms have an exponential running time for non-colorable graph instances. Upper bounds for the resolution complexity can also be established by analyzing some typical backtracking heuristics [21]. For edge probability $p = \frac{c}{n}$ with c sufficiently large, an expected polynomial-time algorithm has been proposed. The algorithm is based a polynomial-time approximation scheme for the *vector chromatic number* of a graph [104].

2.4.3 Hamiltonian Cycle

The threshold behavior of the Hamiltonian cycle problem in random graphs is well understood and can be summarized as follows.

Theorem 2.4.2 (See [30]). *Let $G(n,p)$ be a random graph with $p = \frac{1}{n}(\log n + \log\log n + c_n)$. Then,*

$$\lim_n \Pr\{G(n,p) \text{ is Hamiltonian }\} = \lim_n \Pr\{\delta(G(n,p)) \geq 2\} = \begin{cases} 0, & \text{if } c_n \to -\infty \\ e^{-e^{-c}}, & \text{if } c_n \to \text{ a constant } c \\ 1, & \text{if } c_n \to +\infty, \end{cases}$$

where $\delta(G(n,p))$ is the minimum vertex degree.

The first breakthrough in the study was made by Pósa in 1976 (see, e.g., [30]) who proved that the random graph $G(n,p)$ is Hamiltonian **whp** if $p \geq \frac{K\log n}{n}$ and $K \geq 16$. A key concept in Pósa's proof is the *path rotation*. Given a path $P = \{v_0, \cdots, v_k\}$, a path rotation $ROTATE(P, v_k, v_i)$ is a new path $\{v_0, \cdots, v_i, v_k, v_{k-1}, \cdots, v_{i-1}\}$ provided that (v_k, v_i) is an edge. Let X' be the set of vertices each of which is an endpoint of a path obtained by a sequence of path rotations starting from P and using v_0 as the fixed endpoint. A key step in Pósa's proof is to show that for any longest path P in a random graph $G(n,p)$, the size of the subset X' must be "large". This is possible because of the facts that (1) the size of the open neighborhood of X' is less than $2|X'|$; and (2) **whp** $G(n,p)$ does not contain any small-sized vertex subset S whose open-neighborhood has a size less than $2|S|$. Insight obtained in the analysis of the threshold behavior of the Hamiltonian cycle problem has motivated several average or **whp** polynomial-time algorithms [30].

The typical-case complexity of backtracking algorithms for the Hamiltonian cycle problem was studied in [134]. Unlike random SAT and random graph k-colorability that have a typical-case complexity peak at the phase transition, it is shown in [134] that the probability of generating hard Hamiltonian cycle instances at phase transitions is extremely low. In fact, backtracking algorithms equipped with pruning techniques specially designed to exploit the unique characteristics of the Hamiltonian cycle problem typically have a linear running time [134].

2.4.4 Number Partitioning

The number partitioning problem described in Problem 2.1 asks whether a given set of integers has a *perfect partitioning*, i.e., a partitioning in which the sums of the two subsets are equal. A more general problem is to ask whether the absolute difference between the sums of the two subsets, called the *discrepancy*, is less than a given value. The so-called *constrained number partitioning problem* is also of interest that imposes a constraint on the difference between the cardinalities of the two subsets in a partitioning [32].

Random Models and Control Parameters

To come up with a random model for the number partitioning problem is straightforward—a random instance is simply a set of n integers drawn randomly from a specified range. It is, however, not so obvious what parameter one should look at in order to identify a phase transition of the solution probability. The lack of an understanding of the nature of the control parameters has led to an incorrect conclusion that the random number partitioning problem has "no phase transition of any kind" until the work of Gent and Walsh [76] who identified the correct control parameter. A random instance of the number partitioning problem is a pair $(\mathbf{X}, M)$ where $\mathbf{X}$ is a set of integers $\{X_1, \cdots, X_n\}$ chosen independently and uniformly from the set of all the integers less than or equal to a given integer M. The parameter proposed in [76] is the ratio κ defined as

$$\kappa = \frac{\log_2 M}{n}.$$

Notice that $\lceil \log_2 M \rceil$ is the maximum number of bits required to represent an integer in the instance. Another way to describe the random number partition problem is to consider a set of n real-valued numbers chosen independently and uniformly from $(0, 1)$ and to use as the control parameter the ratio between n and the maximum effective number of digits.

Phase Transition of Solution Probability

The phase transition of the solution probability of the random number partitioning problem is well characterized. Let $Z_{n,l}$ be the number of partitions with discrepancy l and write $Z_n = Z_{n,0}$ for the number of perfect partitions. Based on estimations of the first and second moments of $Z_{n,l}$, the following threshold $\kappa_c = 1$ was established [33]

$$\lim_n \Pr\{ Z_n > 0 \} = \begin{cases} 1, & \text{if } \frac{\log_2 M}{n} < 1 \\ 0, & \text{if } \frac{\log_2 M}{n} > 1 \end{cases}$$

In fact, a simple induction on n shows that the following representation of $Z_{n,l}$ is correct:

$$Z_{n,l} = 2^n I_{n,l} \times \begin{cases} 1, & \text{if } l = 0 \\ 2, & \text{if } l > 0 \end{cases}$$

where

$$I_{n,l} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \cos(lx) \prod_{j=1}^{n} \cos(x \times X_j) dx.$$

It follows from the independence of $\{X_i, 1 \leq i \leq n\}$ and the Fubini theorem that

$$\mathcal{E}[I_{n,l}] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \cos(lx) \mathcal{E}^n[\cos(x \times X)] dx$$

where X is a random variable uniformly distributed on the set of integers $\{1, 2, \cdots, M\}$. And therefore, $\mathcal{E}^n[\cos(x \times X)]$ has an explicit expression. Explicit formulas for $\mathcal{E}[I_{n,l_1} I_{n,l_2}]$ can be derived similarly. A detailed analysis can be found in [33]. The phase transition of the constrained number partitioning problem was analyzed in [32].

Algorithms and Typical-case Complexity

The number partitioning problem has a pseudo-polynomial time algorithm based on standard dynamic programming techniques. In fact, the algorithm's running time is polynomial if the sum of the numbers to be partitioned is polynomial in n—the cardinality of the set of the numbers to be partitioned.

In [76], several heuristic algorithms were empirically analyzed in terms of the phase transition behavior of the random number partitioning problem, including *Korf's greedy heuristic*, *the set-difference heuristic*, and a backtracking algorithm called the *CKK algorithm*. In Korf's greedy heuristic, numbers are iteratively put into one of the two subsets. In each iteration, the largest remaining number is selected and added to the subset with the smaller sum. The set-difference heuristic recursively replaces two numbers by their difference. In effect, this is equivalent to asserting that the two numbers should be in different subsets; The Karmarkar and Karp heuristic (KK heuristic) is a special set-difference heuristic that always chooses the two largest numbers to replace; The CKK algorithm is a backtracking algorithm that uses the KK heuristic to branch. The pseudo-code of the CKK algorithm is given in Algorithm 2.4.

Algorithm 2.4 CKK algorithm for number partitioning

```
CKK(X)
Input: A set X of integers.
if (|X| ≤ 4) then
  return SUCCESS/FAIL accordingly
end if
Let x, y be the two largest numbers in X and assume x ≥ y
Let a ← (x − y) and b ← (x + y)
if (CKK((X \ {x, y}) ∪ {a}) == SUCCESS) then
  return SUCCESS
else
  return CKK((X \ {x, y}) ∪ {b}))
end if
```

Another interesting empirical work is a study on the relation between the phase transition and the shape of the energy landscape of the optimal number partition problem [130]. The statistics of the shape of the landscape, such as the number of local minima and the height of the barriers, are summarized by a so-called barrier tree. The conclusion drawn in [130] is that except for the

maximum barrier height, almost all of the other features considered remain constant across the threshold and thus, are insensitive to the phase transition.

2.5 Generating Hard Instances

Analytical approaches, while being rigorous, often fail to tell "the full story about real-world algorithmic performance" [93] because of the worst-case nature of the results or the over-simplified assumptions made in order to make mathematical treatment amenable. As a result, empirical study remains an important approach to the analysis of algorithms [93].

In empirical studies, a specific set of problem instances is used as test instances to gain insight on the strengths and weaknesses of the algorithms under consideration and/or to reveal some intrinsic characteristics of the algorithmic problems under investigation. In addition to benchmarks manually compiled from real-world applications, random problem instances are also widely used. However, generating (hard) random instances with controlled characteristics is not a trivial task. A classical example is a random SAT model used in 1980s [78] that has been shown to have an extremely strong bias towards generating, sometimes trivially, easy instances [62]. A more recent example is a class of widely-used CSP models which have been proved to be trivially unsatisfiable asymptotically with probability one [8]. In fact, the existence of a non-deterministic polynomial-time program to construct all the instance and optimal solution pairs of an NP-hard optimization problem with some specified characteristics will imply that NP = co-NP [126].

Selman, et al. [127] were perhaps the first to recognize the potential and importance of using (hard) instances generated at phase transitions as benchmark problems. The study of phase transitions and typical-case complexity of random models of NP and co-NP problems provide a valuable guidance to the design of random instance generators. Even though randomly-generated instances can be easily criticized for their lack of the structures that frequently appear in real-world problems, they are in fact one of the driving forces behind the recent dramatic performance improvement of SAT solvers. Nonetheless, there have been several proposals that generate test instances for satisfiability by encoding random problem instances from other domains such as the quasi-group completion problem [79] and the subgraph-isomorphism problem, or by "morphing" random instances from several different domains [74].

As has been discussed previously, random instances generated from above the phase transition threshold typically have an exponential resolution complexity and thus, are hard for many complete algorithms such as backtracking. The generating of hard satisfiable random instances is, however, a challenging task. A straightforward but not very efficient method is to first generate random instances at the phase transition and then filter out those unsatisfiable instances by some complete search algorithm. In the past decade, efforts have

been focused on generating satisfiable instances by embedding (or hiding) a pre-specified solution.

Forced-formula in SAT

A forced-formula [15] is a random CNF formula satisfying a pre-specified assignment. A plain forced-formula is generated as follows:

1. Randomly generate a truth assignment T;
2. Randomly generate a clause. Add the clause to the formula only if it is satisfied by T;
3. Repeat Step 2 until the formula contains the required number of clauses.

A drawback of the plain forced-formula is that in addition to the pre-specified solution, they usually have many "by-product" solutions that create a strong enough statistical bias for randomized local search algorithms to exploit. The generator AIM developed in [15] can generate not only plain forced-formulas but also forced-formulas with a unique or a small number of solutions. The idea used in the plain forced-formula scheme has been extended in [19, 7]. Instead of hiding one solution, the authors in [19, 7] proposed to hide two complementary solutions. This can be implemented as follows. Let $T = \{t_1, \cdots, t_n\}$ be an assignment and $\overline{T} = \{1 - t_1, \cdots, 1 - t_n\}$ be the complement. To generate a random clause of size k, we first randomly select a set of k variables $\{x_{i_1}, \cdots, x_{i_k}\}$. Then, we select according to some distribution a clause from the set of $2^k - 2$ possible clauses on $\{x_{i_1}, \cdots, x_{i_k}\}$ that are satisfied by both T and $\overline{T}$. In [7], each of the $2^k - 2$ potential clauses is selected with equal probability, while in [19], the distribution is defined by two parameters that relate the probability of selecting a potential clause to its Hamming distance to the pair of pre-specified complementary assignments on $\{x_{i_1}, \cdots, x_{i_k}\}$.

Hidden-color in Graph Coloring

The basic idea behind generating random k-colorable graphs is as follows. First, the set of vertices V is partitioned into k *color-classes* $\mathbf{C} = \{V_i, 1 \leq i \leq k\}$. Then edges are selected according to some distribution from the set of potential edges $\{(u, v) \in V^2 : u \in V_i, v \in V_j, i \neq j\}$.

It is easy to see that a graph generated in this way is guaranteed to have at least one valid k-coloring. Variants of random models of k-colorable graphs can be defined by specifying how the color-classes are formed and how the edges are selected. These models can be classified into two categories, the *random k-colorable graph* and the *semi-random k-colorable graph*.

Random k-Colorable Graphs $G(n,p,k)$

In $G(n,p,k)$, after the partitioning $\mathbf{C} = \{V_i, 1 \leq i \leq k\}$ is specified, each pair of vertices in $\{(u,v) \in V^2 : u \in V_i, v \in V_j, i \neq j\}$ is selected as an edge independently and with probability p. Below are several widely used partitioning schemes:

1. **Equi-Partite [104]**. In this scheme, the vertex set is partitioned into color-classes of equal size.
2. **Uniform-Partite**. In this scheme, each vertex is assigned to one of the k color-classes uniformly and independently.
3. **δ-Partite [43]**. The variation of the size of the color-classes can be controlled by some parameter δ in several different ways.

Semi-Random k-Colorable Graphs

In the semi-random model, additional structures and restrictions are used when selecting edges of the graph.

1. $G_s(n,k,p)$**[132]**. In this model, for each pair of vertices (u,v) from different color-classes, an adversary picks a value $p_{uv} \in [p,1]$ and includes (u,v) as an edge with probability p_{uv}.
2. **Flat Graph [43].** In the flat graph, the color-classes are of equal size, but the edges are selected in such way that for each pair of color classes V_i and V_j, the maximum vertex degree of the bipartite subgraph on (V_i, V_j) is upper bounded by a pre-specified constant.

Quasigroup with Holes [6]

A *quasigroup* is a pair $(Q,*)$ where Q is a finite set of symbols and $* : Q \times Q \rightarrow Q$ is a binary operation such that equations of the form

$$a * x = b, \text{ and } y * a = b$$

have a unique solution. The multiplication table of a quasigroup defines a *Latin square*, i.e., a $|Q| \times |Q|$ table of symbols from Q such that no orthogonal row or column contains the same symbol from Q twice. $|Q|$ is called the order of the quasigroup.

In the *quasigroup completion problem* (QCP), an instance is a partially-filled multiplication table and the question is to decide whether the unfilled entries of this table can be filled to obtain a Latin square. The quasigroup completion problem has recently been used to generate structured random test instances for CSP and SAT search algorithms [80, 74]. In an effort to design

better satisfiable random instance generators, Achlioptas, et al. [6] proposed the *quasigroup with holes (QWH)* problem. A random instance of QWH is generated in two steps: (1) using the Markov Chain Monte Carlo algorithm to generate a uniformly-distributed Latin square; and (2) deleting a fraction of entries of the table ("punching some holes in the table"). The resulting table is an instance of QWH. In fact, the instance is a satisfiable instance of QCP.

Experiments conducted in [6] showed that random QWH has interesting phase transitions and associated easy-hard-easy complexity patterns for both complete and incomplete search algorithms.

Chapter 3

Constraint Satisfaction Problem and its Random Models

This chapter is an overview of the constraint satisfaction problem, its random models, and previous work on the phase transitions in these models.

3.1 Constraint Satisfaction Problem

The study of constraint satisfaction problems was initiated by Montanari in his work on problems in image processing [119, 106]. It turns out that the idea of constraint satisfaction can be used to model a large variety of problems.

In a constraint satisfaction problem, we are given a set of variables and a collection of *constraints*. Each constraint is defined over a fixed subset of variables and specifies a set of value-tuples that these variables can simultaneously take. The task is to find an assignment to the variables to satisfy all the constraints.

Definition 3.1.1 (Variables, domains, and projections). *Consider a set of variables* $X = \{x_1, \cdots, x_n\}$ *where each variable* x_i *takes its value on a finite set* Ω_i*, called its* domain. *We use*

$$\mathbf{\Omega} = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$$

to denote the space of all the n-tuples that X *can take. If all the variables have the same domain* Ω*, we will simply write* $\mathbf{\Omega} = \Omega^X = \Omega^n$*. Alternatively, each variable* x_i *can be viewed as a* projection function $x_i(\omega) : \mathbf{\Omega} \to \Omega_i$ *defined as*

$$x_i(\omega) = \omega_i, \quad \textit{where } \omega = (\omega_1, \cdots, \omega_n) \in \mathbf{\Omega}.$$

Definition 3.1.2 (Constraints, restrictions, nogoods, and support). *A* ***constraint*** C *is a relation defined on the product space of the domains of a subset of variables. The subset of variables is called the scope of the constraint and is denoted by* $\mathrm{var}(C)$*. The arity of the constraint is the number*

of variables in its ***scope****. In particular, a constraint of arity 2 is called a* ***binary constraint****.*

The constraint relation of a constraint C can be identified as a subset

$$\{\omega \in \Omega^{\mathbf{var}(C)} : C(\omega) = 0\}$$

of the product space of the domains of its scope variables. Without causing confusion, we will view a constraint as a relation or a subset on the product space of domains interchangeably.

A value-tuple $(\omega_{i_1}, \cdots, \omega_{i_k})$ is said to be incompatible if $C(\omega_{i_1}, \cdots, \omega_{i_k}) = 0$. Otherwise it is said to be compatible. An incompatible value-tuple is also called a ***restriction*** *or a* ***nogood*** *of the constraint. The* ***tightness*** *of a constraint C is defined to be the number of nogoods of C.*

Let x be a variable and ω_x be a domain value of x. A ***support*** *for ω_x from another variable y is a domain value ω_y of y such that (ω_x, ω_y) is compatible. If there is no constraint between x and y, then any domain value of y is a support of any domain value of x.*

Definition 3.1.3 (Constraint Satisfaction Problem). *A constraint satisfaction problem can be formally stated as follows.*

> ***Constraint Satisfaction Problem (CSP)***
> INSTANCE: *A set of variables, their domains, and a collection $\mathcal{C}$ of constraints.*
> QUESTION: *Is the set $\bigcap_{C \in \mathcal{C}} C^{-1}(1)$ non-empty?*

Definition 3.1.4 (Flawed constraint and flawed variable). *A constraint is said to be flawed if every value-tuple of the scope variables is incompatible. A variable x in a CSP instance is said to be flawed if for each domain value δ of x, there exists a variable y such that δ has no support from y.*

Definition 3.1.5 (Arc-consistent, path-consistent, and generalized arc-consistent). *A constraint C with scope $\{x_{i_1}, x_{i_2}, \cdots, x_{i_k}\}$ is generalized arc-consistent if for any domain value $\overline{\omega}$ of any variable x_{i_j}, there is a compatible value-tuple $(\omega_{i_1}, \omega_{i_2}, \cdots, \omega_{i_k})$ with $\omega_{i_j} = \overline{\omega}$. In particularly, a generalized arc-consistent binary constraint is simply called an arc-consistent constraint.*

A CSP instance is said to be generalized arc-consistent if each of its constraints is generalized arc-consistent.

A binary CSP instance is said to be path-consistent if for 2 variables $\{x_{i_1}, x_{i_2}\}$, any compatible value-tuple $(\omega_{i_1}, \omega_{i_2})$ of $\{x_{i_1}, x_{i_2}\}$, and for any other variable x_{i_3}, there is a domain value ω_{i_3} of x_{i_3} such that $(\omega_{i_1}, \omega_{i_3})$ is a compatible value-tuple of $\{x_{i_1}, x_{i_3}\}$ and $(\omega_{i_2}, \omega_{i_3})$ is a compatible value-tuple of $\{x_{i_2}, x_{i_3}\}$.

Associated with a CSP instance are several graphs that describe the interaction structure of the variables.

Definition 3.1.6 (Constraint (Hyper)Graph and Constraint Primal Graph). *Let $(X, \mathcal{C})$ be a CSP instance where $X = \{x_i, 1 \leq i \leq n\}$ is a set of variables and $\mathcal{C}$ is a collection of constraints.*

1. *The **primal graph** of the CSP instance is a graph $G = G(V, E)$ where V corresponds to the set of variables X and $(v_i, v_j) \in E$ if and only if the corresponding variables x_i and x_j both appear in a constraint in $\mathcal{C}$.*

2. *The **constraint (hyper)graph** is a (hyper)graph $\mathcal{G}(V, E)$ where V corresponds to the set of variables X and E contains all the subsets of variables that correspond to the scope $\mathrm{var}(C)$ of a constraint $C \in \mathcal{C}$.*

Many specific problems can be formulated as a CSP by specifying a set of **constraint templates**—canonical constraints defined on a set of generic variables. From now on, we will assume that all the variables have the same domain Ω. A k-ary constraint template is a boolean function $T : \Omega^k \to \{0, 1\}$. The set of all the possible k-ary constraint templates is denoted by $\mathbb{T}_k$. This is exactly the set of all the Boolean-valued functions defined on Ω^k, and therefore if $|\Omega| = d$, we have $|\mathbb{T}_k| = 2^{d^k}$. We denote by

$$\mathbb{T}^* = \bigcup_{k \geq 1} \mathbb{T}_k$$

the set of all the constraint templates.

Given a k-ary constraint template $T \in \mathbb{T}_k$, a constraint C with the scope $\mathrm{var}(C) = \{x_{i_1}, \cdots, x_{i_k}\}$ can be derived as follows:

$$C(\omega) = T(x_{i_{\pi(1)}}(\omega), \cdots, x_{i_{\pi(k)}}(\omega)), \quad \omega \in \Omega^n,$$

where π is a permutation over $\{1, 2, \cdots, k\}$.

Definition 3.1.7 (Space of CSP Instances). *Let n be the number of variables, T be a constraint template, and $\mathbb{C} \subset \mathbb{T}^*$ be a set of constraint templates. We define the following spaces of CSP instances.*

1. $\mathrm{C}(n, T)$*: the collection of constraints that can be derived from* T.

2. $\mathrm{C}(n, \mathbb{C})$*: the collection of constraints that can be derived from* $\mathbb{C}$.

3. $\mathrm{CSP}_n(\mathbb{C})$*: the collection of the CSP instances whose constraints are from* $\mathrm{C}(n, \mathbb{C})$.

4. $\mathrm{CSP}_{n,m}(\mathbb{C})$*: the subset of* $\mathrm{CSP}_n(\mathbb{C})$ *that have exactly* m *constraints.*

$\mathrm{CSP}_n(\mathbb{C})$ *can also be identified with the product space* $\{0, 1\}^{\mathrm{C}(n,\mathbb{C})}$ *since each vector* $(L_C, C \in \mathrm{C}(n, \mathbb{C})) \in \{0, 1\}^{\mathrm{C}(n,\mathbb{C})}$ *defines a set of constraints* $\{C \in \mathrm{C}(n, \mathbb{C}) : L_C = 1\}$.

The following are some examples of subsets of constraint templates that define some special spaces of CSP instances each of which corresponds to a specific NP-complete decision problem.

Example 3.1.1. *Consider a set of Boolean variables $X = \{x_1, \cdots, x_n\}$. Let $\mathbb{C} \subset \mathbb{T}_k$ be the set of k-ary constraint templates.*

1. *If $\mathbb{C} = \{T_i \in \mathbb{T}_k, 0 \leq i \leq k\}$ where $T_i(\omega) = 0$ iff $\omega = 1^i 0^{k-i}$, then $\mathrm{CSP}_n(\mathbb{C})$ = k-SAT. This is because each T_i is equivalent to the k-clause $\overline{x}_1 \vee \cdots \vee \overline{x}_i \vee \vee x_{i+1} \cdots \vee x_k$.*

2. *If $\mathbb{C} = \{T_i \in \mathbb{T}_k, 0 \leq i \leq k\}$ where $T_i(\omega) = 0$ iff $\omega = 0^i 1^{k-i}$ or $\omega = 1^i 0^{k-i}$, then $\mathrm{CSP}_n(\mathbb{C})$ is equivalent to the Not-All-Equal-SAT problem (NAE-SAT).*

3. *If $\mathbb{C} = \{T_i \in \mathbb{T}_k, 0 \leq i \leq k\}$ where $T_i(\omega) = 0$ iff $(1 - \omega_1) + \cdots + (1 - \omega_i) + \omega_{i+1} + \cdots + \omega_k = 1$, then $\mathrm{CSP}_n(\mathbb{C})$ is the 1-in-k-SAT problem.*

4. *If $\mathbb{C} = \{T_1, T_2\}$ where $T_1(\omega) = \omega_1 \oplus \cdots \oplus \omega_k$ and $T_2(\omega) = \omega_1 \oplus \cdots \oplus \omega_k \oplus 1$, then $\mathrm{CSP}_n(\mathbb{C})$ is the k-XOR-SAT problem.*

Example 3.1.2. *Consider a set of variables $X = \{x_1, \cdots, x_n\}$.*

1. *Assume that each x_i has the domain $\{1, 2, \cdots, k\}$. Let $\mathbb{C} = \{T\} \subset \mathbb{T}_2$ such that $T(\omega) = 1$ iff $\omega_1 \neq \omega_2$. Then, $\mathrm{CSP}_n(\mathbb{C})$ is the graph k-colorability problem.*

3.2 Random Models of Constraint Satisfaction Problems

Generally speaking, a random model of CSPs can be viewed as a probability distribution defined on a space of CSP instances.

Definition 3.2.1 (Random CSP Models). *Given a set of constraint templates $\mathbb{C}$, a random CSP model is a (discrete) probability space $\{\mathrm{CSP}_n(\mathbb{C}), \mathcal{P}\}$ where $\mathcal{P}$ is a probability distribution on $\mathrm{CSP}_n(\mathbb{C})$.*

One way to define a random CSP model is to work directly with the product space $\mathrm{CSP}_n(\mathbb{C})$. For example, we can have the following random models for CSPs.

Definition 3.2.2 (Model $\mathcal{M}_{n,p}^{k,d}$: d—domain size, p—probability, k—constraint arity, n—number of variables). *In this model, each constraint in $C(n, \mathbb{T}_k)$ is selected independently with probability p. This is exactly the random CSP model $(\mathrm{CSP}_n(\mathbb{T}_k), \mathcal{P})$ where $\mathcal{P}$ is the independent product Bernoulli distribution on $\mathrm{CSP}_n(\mathbb{T}_k)$ such that $\forall L = \mathrm{CSP}_n(\mathbb{T}_k)$,*

$$\mathcal{P}(L) = \prod_{C \in C(n, \mathbb{T}_k)} p^{L_C} (1-p)^{1-L_C}.$$

Definition 3.2.3 (Model $\mathcal{M}_{n,m}^{k,d}$). *In this model, a random instance contains exactly m constraints selected uniformly without replacement from $C(n, \mathbb{T}_k)$.*

A potential problem with the above random models is that a random instance may contain two constraints with the same scope. However, it can be proved that for parameters that are in the range of interest, **whp** such events do not occur except for the case of $k = 2$.

Lemma 3.2.1. *Let A_n be the event that $\mathcal{M}_{n,p}^{k,d}$ (or $\mathcal{M}_{n,m}^{k,d}$) contains two constraints with the same scope variables. Then, we have for $p = \Omega(\frac{1}{n^{k-1}})$ (respectively, $m = \Omega(n)$),*

$$\lim_n \Pr\{A_n\} \begin{cases} > 0, & \text{if } k = 2; \\ = 0, & \text{if } k \geq 3 \end{cases}$$

Proof. We only consider the model $\mathcal{M}_{n,p}^{k,d}$. Let $a = |\mathbb{T}_k|$ (a constant). For a fixed set S of variables, the number of constraints with the scope S follows the binomial distribution $b(a, p)$. Thus, the probability p_S that $\mathcal{M}_{n,p}^{k,d}$ contains more than one constraint with the scope S is

$$p_S = \sum_{i=2}^{a} \binom{a}{i} p^i (1-p)^{a-i} = O(p^2).$$

The number of subsets of variables on which $\mathcal{M}_{n,p}^{k,d}$ contains more than one constraint has a binomial distribution $b(n^k, p_S)$. Therefore, we have

$$\Pr\{A_n\} = 1 - (1 - p_S)^{n^k},$$

and the lemma follows. □

A more intuitive way to specify random models of CSPs is as follows. First, a random graph (or random hypergraph) is used as the constraint graph. Then, for each edge of the constraint (hyper)graph, randomly select a constraint that uses the constraint edge as its scope. This approach in its most general form is summarized in [115].

Definition 3.2.4 (The model $\mathcal{CSP}_n^{d,k}(G, \mathbb{P}) = \{\mathrm{CSP}_n(\mathbb{T}_k), \mathcal{P}\}$). *In this model, the distribution $\mathcal{P}$ is specified by a pair $(G, \mathbb{P})$ where G is a k-homogeneous random hypergraph and $\mathbb{P}$ is a probability distribution on the $\mathbb{T}_k$.*

We use $\mathcal{CSP}_{n,m}^{d,k}(\mathbb{P})$ (or $\mathcal{CSP}_{n,p}^{d,k}(\mathbb{P})$) to denote the corresponding model in which the constraint hypergraph is the k-homogeneous uniform random hypergraph $\mathcal{G}^k(n, m)$ (respectively, the k-homogeneous constant-probability random hypergrapg $\mathcal{G}^k(n, p)$).

A random instance of $\mathcal{CSP}_n^{d,k}$ is generated as follows. First, a random hypergraph is generated. Then for each hyperedge, a constraint on the hyperedge is derived from a constraint template $T \in \mathbb{T}_k$ where T is selected from $\mathbb{T}_k$ according to the distribution $\mathbb{P}$.

3.2.1 Classical Random CSP Models

All of the classical random CSP models are a special case of $\mathcal{CSP}_n^{d,k}(G, \mathbb{P})$. In these models the constraint hypergraph is either the constant-probability random hypergraph $\mathcal{G}^k(n,p)$ or the uniform random hypergraph $\mathcal{G}^k(n,m)$. The nogoods of a constraint are determined by either (1) uniformly selecting a subset of value-tuples from all the possible value-tuples of the given arity; or (2) choosing each value-tuple with a fixed probability. Four different combinations give us four classical random CSP models known as Models A, B, C, and D [75].

Definition 3.2.5 (Model $\mathcal{A}_{n,p}^{d,k,q}$).

Parameters: n—number of variables; p—edge probability; d—domain size; k—constraint arity; q—nogood probability.

Distribution: The same distribution as $\mathcal{CSP}_{n,p}^{d,k}(\mathbb{P})$ where the probability distribution $\mathbb{P}$ is defined on the set $\mathbb{T}_k$ of all the possible constraint templates of arity k such that for any $T \in \mathbb{T}_k$ with tightness $0 \leq t \leq d^k$,

$$\mathbb{P}(T) = q^t(1-q)^{d^k - t}.$$

Comments: It is obvious that with non-zero probability, there is a constraint in Model A that excludes all the possible value-tuples, resulting in a trivially unsatisfiable instance.

Definition 3.2.6 (Model $\mathcal{B}_{n,m}^{d,k,t}$).

Parameters: n—number of variables; m—number of constraints; d—domain size; k—constraint arity; t—tightness.

Distribution. The same distribution as $\mathcal{CSP}_{n,m}^{d,k}(\mathbb{P})$ where $\mathbb{P}$ is the uniform distribution on

$$\mathbb{T}_k^t = \{T \in \mathbb{T}_k : \text{ The tightness of } T \text{ is } t\}$$

Definition 3.2.7 (Model $\mathcal{C}_{n,p}^{d,k,t}$).

Parameters: n—number of variables; p—edge probability; d—domain size; k—constraint arity; t—tightness.

Distribution. The same distribution as $\mathcal{CSP}_{n,p}^{d,k}(\mathbb{P})$ where $\mathbb{P}$ is the uniform distribution on

$$\mathbb{T}_k^t = \{T \in \mathbb{T}_k : \text{ The tightness of } T \text{ is } t\}$$

Definition 3.2.8 (Model $\mathcal{D}_{n,p}^{d,k,t}$).

Parameters: n—*number of variables;* m—*number of constraints;* d—*domain size;* k—*constraint arity;* q—*nogood probability.*

Distribution. *The same distribution as* $CSP^{d,k}_{n,m}(\mathbb{P})$ *where the probability distribution* $\mathbb{P}$ *is defined on the set* $\mathbb{T}_k$ *of all the possible constraint templates of arity* k *such that for any* $T \in \mathbb{T}_k$ *with tightness* $0 \leq t \leq d^k$,

$$\mathbb{P}(T) = q^t(1-q)^{d^k - t}.$$

3.2.2 Improved Random CSP Models

CSP is one of the NP-complete problems whose phase transition has been studied since the early 90's. Stimulated by the work of Achlioptas et al [8] showing the flawedness of the classical random CSP models, there has been growing interest in designing appropriate random models with non-trivial phase transitions and studying the typical-case complexity of these random models.

Achlioptas et al [8] proposed the model E as an alternative to the classical random CSP models.

Definition 3.2.9 (Model E).

Parameters: n—*number of variables;* d—*domain size;* k—*constraint arity;* m—*number of nogoods.*

Distribution: *For each set of variables of size* k, *there are* d^k *possible nogoods. In model E, we selects independently and uniformly with replacement* m *nogoods from the* $\binom{n}{k}d^k$ *possible nogoods of all the possible subsets of variables of size* k. *Once the* m *nogoods are determined, we form a CSP instance by merging nogoods on the same set of variables into a single constraint.*

Comments: *It has been proved that model E with* $m = \Theta(n)$ *has interesting threshold behaviors. However, as we are going to show, model E with* $m = \Theta(n)$ *is not without problem—the constraint tightness of an instance is* **whp** *less than 2. This indicates that model E is simply a "multi-valued" version of the Boolean SAT model, and thus, short of many features that we expect to see in CSPs.*

Lemma 3.2.2. *Assume that* $m = \Theta(n)$ *and let* A_n *be the event that all the constraints in a random instance of model E have a constraint tightness less than 2. Then, we have*

$$\lim_n \Pr\{A_n\} = 1, \text{ for any } k \geq 3.$$

Proof. Let A^i_n be the event that the first i nogoods are on different subsets of variables, and B^i_n be the event that the i-th nogood is on a subset of variables

different from those of the previous $(i-1)$ nogoods. We have

$$\begin{aligned}\Pr\{A_n\} &= \Pr\{B_n^m|A_n^{m-1}\} \times \Pr\{B_n^{m-1}|A_n^{m-2}\} \cdots \Pr\{B_n^2|A_n^1\} \\ &= \prod_{i=1}^{m-1}\left(1-\frac{i}{\binom{n}{k}}\right) \\ &\geq \left(1-\frac{m-1}{\binom{n}{k}}\right)^{m-1}\end{aligned}$$

The lemma follows. □

Gent, et al [75] proposed the *flawless model* to overcome the triviality of the classical random CSP models. A key observation is that the existence of the flawed variables might be a direct result of the fact that classical random CSPs are not arc-consistent. A more detailed discuss on the flawless model will be given in Chapter 5 where a random CSP model generalizing this idea will be developed.

Another viable approach to overcoming the triviality is to allow a slowly increasing domain size in the classical random CSP models. Threshold behaviors and typical-case complexity of such a CSP models have been discussed in [58, 129, 140].

3.3 Phase Transitions of Random CSPs

As we have discussed in Chapter 2, there is a phase transition of the solution probability in random models of several NP-complete problems such as k-SAT and graph k-colorability. In these problems, the threshold behavior is largely determined by a **single** parameter that summarizes the "constraintedness" of the random models. For random k-SAT, the parameter is the clause density of the formula, while for graph k-colorability, it is the edge density of the random graph. This is, however, far from true for general CSPs. In fact, local properties specifying how the nogoods of the constraints are selected play a significant role in the threshold behavior of general random CSPs.

3.3.1 Flawed Variables, Flawed Constraints, and Well-behaved CSPs

Achlioptas, et al [8] were the first to notice the difference between general CSPs and other special problems such as k-SAT. It was shown in [8] that in most regions of the parameter space, the four classical random CSP models A-D are trivially unsatisfiable in the sense that **whp** they all contain "*flawed variables*" (variables whose domain values are all incompatible with some other variables) or "*flawed constraints*" (constraints that contain all the possible value-tuples

as its nogoods). As a consequence, there is in fact no phase transition in these models at all. Results related to the phenomena of trivially unsatisfiability in random CSPs can be summarized in the following theorem.

Theorem 3.3.1 ([8, 75]). *Let $q > 0$ and $c > 0$ be fixed constants.*

1. *For any m and n s.t. $\frac{m}{n} = c$ and $p = \frac{c}{n^{k-1}}$.*

$$\lim_n \Pr\{\, \mathcal{A}^{d,k,q}_{n,p} \text{(or } \mathcal{D}^{d,k,q}_{n,m}\text{) has a flawed constraint} \,\} = 1.$$

2. *For any m and n s.t. $\frac{m}{n} = c$, $p = \frac{c}{n^{k-1}}$, and $t \in [d^{k-1}, d^k]$,*

$$\lim_n \Pr\{\, \mathcal{B}^{d,k,t}_{n,m} \text{ (or } \mathcal{C}^{d,k,t}_{n,p}\text{) has a flawed variable} \,\} = 1.$$

3. *For any m and n s.t. $\frac{m}{n} < \frac{1}{k(k-1)}$, $p = \frac{(k-2)!}{n^{k-1}}$ and $t \in [0, d^{k-1})$,*

$$\lim_n \Pr\{\, \mathcal{B}^{d,k,t}_{n,m} \text{ (or } \mathcal{C}^{d,k,t}_{n,p}\text{) is satisfiable} \,\} = 1.$$

Proof. (1) In $\mathcal{A}^{d,k,q}_{n,p}$ and $\mathcal{D}^{d,k,q}_{n,m}$, the number of nogoods of a constraint has the binomial distribution $b(d^k, q)$. The probability for a constraint to have all the d^k possible tuples as its nogoods is thus q^{d^k}. Since the nogoods of the m constraints are determined independently, the probability that there is at least one flawed constraint is

$$1 - (1 - q^{d^k})^{cn}.$$

(2) We only consider $\mathcal{C}^{d,k,t}_{n,p}$ with $k = 2$. Let G be the constraint graph of $\mathcal{C}^{d,2,t}_{n,p}$. Fixing a variable ordering $\{x_1, \cdots, x_n\}$ and let $V_i = \{x_i, \cdots, x_n\}, 1 \leq i \leq n$. Consider a procedure that at each step $1 \leq i \leq \frac{n}{2}$, checks to see if the variable x_i is flawed in the sub-instance induced by V_i. Let p_i be the probability that at step i, the procedure finds that x_i is flawed in the sub-instance induced by V_i. Let A_i be the event that the vertex degree of x_i in the induced graph G_{V_i} on V_i is exactly d, and let p_f be the probability that a variable with d incident constraints is flawed. Then, we have that $p_f > 0$. Since $p = \frac{c}{n}$, it is also true that

$$\Pr\{\, A_i \,\} \geq \Pr\{\, A_{\frac{n}{2}} \,\} > 0.$$

Since the sequence of events $\{A_i, 1 \leq i \leq \frac{n}{2}\}$ are independent, the probability that $\mathcal{C}^{d,k,t}_{n,p}$ has a flawed variable is at least

$$1 - \prod_{i=1}^{n/2} p_f(1 - \Pr\{\, A_i \,\}) > 1 - \left(p_f(1 - \Pr\{\, A_{\frac{n}{2}} \,\})\right)^{\frac{n}{2}} \to 1.$$

(3) If $t \in [0, d^{k-1})$, a randomly-generated instance is always arc-consistent. This, together with the fact that for $\frac{m}{n} < \frac{1}{k(k-1)}$, $p = \frac{(k-2)!}{n^{k-1}}$, the random hypergraphs $\mathcal{G}(n, m)$ and $\mathcal{G}(n, p)$ **whp** contains only hypertrees and unicycle components, finishes the proof. □

Theorem 3.3.1(3) indicates that for $t \in [0, d^{k-1})$, $\mathcal{B}_{n,m}^{d,k,t}$ and $\mathcal{C}_{n,p}^{d,k,t}$ are still interesting. The models $\mathcal{A}_{n,m}^{d,k,q}$ and $\mathcal{D}_{n,p}^{d,k,q}$, however, remain trivial even if $q = q(n) = o(1)$. In fact, an argument similar to those in [70] can be used to show that if $q(n)$ decreases too slowly, the two models are still trivially unsatisfiable, but if $q(n)$ decreases too fast, randomly-generated instances of these two models can be decomposed into subproblems of fixed sizes.

Molloy introduced in [115] the random CSP model $\mathcal{CSP}_{n,m}^{d,k}(\mathbb{P})$ (see Definition 3.2.4). Several general conditions on the support $supp(\mathbb{P}) \subset \mathbb{T}_k$, i.e. the set of constraint templates T such that $\mathbb{P}(T) > 0$, were identified for $\mathcal{CSP}_{n,m}^{d,k}(\mathbb{P})$ to have interesting threshold behaviors.

Definition 3.3.1 ([115]). *The model $\mathcal{CSP}_{n,m}^{d,k}(\mathbb{P})$ is said to have a partial phase transition if there are constants $c_1, c_2 > 0$ such that*

$$\lim_n \Pr\left\{ \mathcal{CSP}_{n,m}^{d,k}(\mathbb{P}) \text{ is satisfiable } \right\} \begin{cases} > 0, & \text{if } m < c_1 n; \\ = 0, & \text{if } m > c_2 n \end{cases}$$

$\mathcal{CSP}_{n,m}^{d,k}(\mathbb{P})$ is said to have a phase transition if there are constants $c_1, c_2 > 0$ such that

$$\lim_n \Pr\left\{ \mathcal{CSP}_{n,m}^{d,k}(\mathbb{P}) \text{ is satisfiable } \right\} = \begin{cases} 1, & \text{if } m < c_1 n; \\ 0, & \text{if } m > c_2 n \end{cases}$$

Let $\mathbb{C} \subset \mathbb{T}^*$ be a set of constraint templates. We say that $\mathbb{C}$ is *symmetric with respect to the domain* Ω [115] if for any two domain values δ and $\gamma \in \Omega$, there is a bijection $\phi : \Omega \to \Omega$ such that for any constraint C, we have $C \in \mathbb{C}$ if and only if the constraint $\phi(C)$ is in $\mathbb{C}$, where $\phi(C)$ is the composition of the relation C and the mapping ϕ. The following conditions are due to Molloy [115], where the case of asymmetric subsets of constraint templates is also discussed.

Definition 3.3.2 ([115]). *Let $\mathbb{C} \subset \mathbb{T}^*$ be a set of constraint templates. We say that $\mathbb{C}$ is well-behaved if it satisfies the following two conditions:*

1. *Any constraint template $C \in \mathbb{C}$ is generalized arc-consistent.*
2. *$\forall \delta \in \Omega$, there is at least one $C \in \mathbb{C}$ s.t. $(\delta, \ldots, \delta)$ is a nogood of C.*

$\mathbb{C}$ is said to be very well-behaved if in addition, we have

3. *a CSP instance is always satisfiable if its constraints are all derived from $\mathbb{C}$ and its constraint hypergraph is a cycle.*

Notice that condition 3 in the above definition is stronger than arc-consistency, but weaker than path-consistency. The following is a generalization of Theorem 3.3.1.

Theorem 3.3.2 ([115]). *$\mathcal{CSP}_{n,m}^{d,k}(\mathbb{P})$ has a partial phase transition (or phase transition) if and only if the support $supp(\mathbb{P})$ of the probability distribution $\mathbb{P}$ is well-behaved (respectively very well-behaved).*

3.3.2 Sharpness of CSP Phase Transitions

The beauty of phase transitions in NP-complete problems lies in the sudden and dramatic jump of the solution probability at a threshold of the model parameter. However, determining the existence and the exact value of such a threshold remains a challenging open problem for some of the NP-complete problems including random k-SAT and k-colorability in random graphs. A breakthrough is Friedgut's work [65] showing that we can study the sharpness of the transition without knowing the existence of the threshold. Random k-SAT, random graph k-colorability, and many other NP-complete problems have been shown to have a sharp phase transition [5, 65, 105].

After the introduction of the general random CSP model $\mathcal{CSP}^{d,k}_{n,m}(\mathbb{P})$ in [115], and independently in [42] in the case of domain size 2, several efforts have been made to identify conditions under which random CSPs have a sharp phase transition [42, 90, 115]. We summarize below the conditions that have been established so far.

Definition 3.3.3 (Sharp Threshold of Random CSPs). *Assume that $\mathcal{CSP}^{d,k}_{n,m}(\mathbb{P})$ has a phase transition. If there is a function $c(n) > 0$ s.t. for any $\epsilon > 0$,*

$$\lim_n \Pr\left\{ \mathcal{CSP}^{d,k}_{n,m}(\mathbb{P}) \text{ is satisfiable } \right\} = \begin{cases} 1, & \text{if } m < (c(n) - \epsilon)n; \\ 0, & \text{if } m > (c(n) + \epsilon)n, \end{cases}$$

we say that $\mathcal{CSP}^{d,k}_{n,m}(\mathbb{P})$ has a sharp threshold at the phase transition. Otherwise, $\mathcal{CSP}^{d,k}_{n,m}(\mathbb{P})$ has a coarse threshold.

It turns out that the notion of a "very well-behaved" collection of constraint templates (Definition 3.3.2) captures much propotion of the conditions for random CSPs to have a sharp threshold. But at least for binary CSPs, it is far from sufficient [84, 115]. Below are some of the currently-known sufficient conditions.

Theorem 3.3.3. *1. [84] Model $\mathcal{B}^{d,k,t}_{n,m}$ has a sharp threshold for any $d, k \geq 2$ and t such that $1 \leq t < d^{k-1}$.*

2. [42, 90] Model $\mathcal{CSP}^{d,k}_{n,m}(\mathbb{P})$ has a sharp threshold when $d = 2$, supp$(\mathbb{P})$ is very well-behaved, and $\mathbb{P}$ is uniform on supp$(\mathbb{P})$.

3.3.3 Random CSPs with (Slowly) Increasing Domain Size

Interestingly, random CSP models with (slowly) increasing domain size behave quite differently from those with bounded domain size. A series of studies showed that for $d = d(n)$ ranging from $\log^{1/2} n$ to $n^\gamma, \gamma > 0$, the classical CSP models do exhibit interesting threshold behaviors [67, 129, 140].

Rather than citing conditions in these works on the necessary growth rate of the domain size (all of which are complicated and messy), we would like to provide a more intuitive account of why an increasing domain size helps avoid the triviality. Take, for example, the model $\mathcal{D}_{n,m}^{d,2,q}$ where $d = d(n)$ is the domain size and $0 < q < 1$ is the nogood probability. Consider a given constraint edge (x_i, x_j). The probability that a value δ of x_i has no support domain values from x_j is q^d. The probability that all the domain values of x_i has support from x_j is at least $1 - dq^d$. Thus, the probability that no variable is flawed is at least

$$(1 - dq^d)^m$$

which, assuming $m = o(n^2)$, tends to 1 as long as, say, $d = 3\log_{1/q}(n)$. As a matter of fact, large deviation bounds for the binomial distribution can even guarantee that with probability exponentially close to 1, any domain value of any variable has a fixed fraction of supports in any constraint.

Chapter 4

Random CSPs with Polynomial Resolution Complexity

4.1 Introduction

In this chapter, we study the resolution complexity of the classical random CSP model $\mathcal{B}^{2,k,t}_{n,m}$ (Definition 3.2.6) with domain size $d = 2$, constraint arity k, and constraint tightness t. We also assume that the constraint density $\frac{m}{n}$ is independent of n. In fact, we will be considering a slightly generalized version of $\mathcal{B}^{2,k,t}_{n,m}$ that allows for a non-integer tightness t:

1. For an integer t, the constraints are constructed as usual.
2. For a non-integer $t = t_0 + \alpha$, where t_0 is an integer and $0 < \alpha < 1$, a constraint is constructed by selecting a random set of t_0 nogoods with probability $1 - \alpha$ or a random set of $t_0 + 1$ nogoods with probability α.

This generalized model is still denoted by $\mathcal{B}^{2,k,t}_{n,m}$ with the understanding that the parameter t can now take any real value in the interval $(0, 2^k]$.

Since the domain size is 2, each constraint with a constraint tightness t corresponds in a straightforward way to a set of t clauses defined on the same set of variables. Consequently, an instance of $\mathcal{B}^{2,k,t}_{n,m}$ is equivalent to a k-CNF formula. Thus, we define the resolution complexity of $\mathcal{B}^{2,k,t}_{n,m}$ to be the resolution complexity of the equivalent CNF formula.

In [8], it is shown that for any $t \in [2^{k-1}, 2^k]$, $\mathcal{B}^{2,k,t}_{n,m}$ is **whp** unsatisfiable because of the existence of the *flawed-variables*. An immediate consequence of this result is that for any $t \in [2^{k-1}, 2^k]$, the resolution complexity of $\mathcal{B}^{2,k,t}_{n,m}$ is almost surely $O(1)$. On the other hand, Mitchell [113] shows that for any $t \in (0, k-1]$, the resolution complexity of $\mathcal{B}^{2,k,t}_{n,m}$ is **whp** exponential.

The main result of this chapter is that for $t \in (2^{k-2}-1, 2^{k-1})$, the resolution complexity of $\mathcal{B}^{2,k,t}_{n,m}$ is **whp** polynomial if the constraint density $\frac{m}{n}$ is high. Specifically, we establish for each $t \in (2^{k-2} - 1, 2^{k-1})$, an upper bound on

the constraint density $\frac{m}{n}$ for $\mathcal{B}_{n,m}^{2,k,t}$ to have an exponential complexity. These upper bounds partly answer the open problem regarding the CSP resolution complexity when the constraint tightness is between $k-1$ and 2^{k-1} [113]. In Section 4.4, we will discuss some more recent and independent work since the publication of the result presented in this chapter [72].

4.2 Main Results

Theorem 4.2.1. *Let $\mathcal{B}_{n,m}^{2,k,t}$ be a random CSP. We have*

$$\lim_{n\to\infty} \Pr\{\, \mathcal{B}_{n,cn}^{2,k,t} \text{ is satisfiable} \,\} = 0$$

if $c = \frac{m}{n}$ and t satisfy one of the following

1. *For $t = 2^{k-2} - 1 + \alpha$ with $0 < \alpha \leq 1$,*

$$c > \frac{\binom{2^k}{2^{k-2}}}{2k(k-1)\alpha} \tag{4.1}$$

2. *For $t = 2^{k-2} + j + \alpha$ with $0 < \alpha \leq 1$ and $0 \leq j \leq 2^{k-1} - 2^{k-2} - 1$,*

$$c > \frac{1}{2k(k-1)} \frac{\binom{2^k}{2^{k-2}}}{\binom{2^{k-2}+j}{2^{k-2}}} (1 + \alpha \frac{2^{k-2}}{j+1})^{-1} \tag{4.2}$$

The theorem is proved by showing that for any constraint tightness t and constraint density c satisfying (4.1) or (4.2), a random instance of $\mathcal{B}_{n,m}^{2,k,t}$ asymptotically almost surely implies an unsatisfiable 2-SAT subproblem. The intuition is that a constraint C with t nogoods is equivalent to a k-CNF formula with t clauses defined on the same set of k variables. If $t > 2^{k-2}$, there is a non-zero probability that these t clauses imply a 2-clause. As a result, if there are enough constraints, we will get enough implied 2-clauses to form an unsatisfiable 2-CNF formula called the criss-cross loop [1]. In fact, this situation has been shown to be true in a different context where the so-called NK landscape model is analyzed ([70]). An NK landscape defined on a set of n variables can be viewed as a special random CSP consisting of exactly n constraints $\{C_1, \ldots, C_n\}$ such that for each $1 \leq i \leq n$, the constraint C_i is defined on the variable x_i and $(k-1)$ other randomly selected variables.

Consider a constraint C_i of $\mathcal{B}_{n,m}^{2,k,t}$. Let $\mathcal{C}_i$ with $|\mathcal{C}_i| = t$ be the set of k-clauses equivalent to C_i and let $\mathcal{F}_i$ be the set of all the 2-clauses that can be derived from $\mathcal{C}_i$. The proof of Theorem 4.2.1 indicates that the set of 2-clauses $\{\mathcal{F}_i, 1 \leq i \leq M = O(m)\}$ is unsatisfiable. Since the resolution complexity of an unsatisfiable 2-SAT problem is polynomial, we have

[1] It should be noted that the implied 2-CNF clauses are not uniformly distributed and the resulting 2-CNF formula is not equivalent to a standard random 2-SAT. Consequently, the current result does not follow from the proof of the satisfiability threshold of the standard random 2-SAT [36, 77].

Theorem 4.2.2. *For any t and $c = \frac{m}{n}$ satisfying the conditions in Theorem 4.2.1, the resolution complexity of $\mathcal{B}_{n,m}^{2,k,t}$ is almost surely linear, and a linear refutation can be obtained in linear time.*

Proof. The set of 2-clauses $\mathcal{F}_i$ can be derived from the set of k-clauses $\mathcal{C}_i$ as follows:

1. Let $\mathcal{D} = \mathcal{C}_i$;
2. Resolve all the pairs of clauses of the form $\{A, x\}$ and $\{A, \overline{x}\}$ in $\mathcal{D}$, where A is a clause of size larger than 2. Insert all the resolvents into $\mathcal{D}$ and repeat this step until there are no more pairs of clauses in $\mathcal{D}$ that can be resolved in this way.
3. Let $\mathcal{F}_i$ be the set of all the 2-clauses in $\mathcal{D}$.

Since the number of constraints is $m = cn$, it takes linear time to run the above procedure for all the constraints, and the length of the resulting sequence of clauses is also linear in n. □

From Theorems 4.2.1 and 4.2.2, we can see that for a given tightness $t \in (2^{k-2} - 1, 2^{k-1})$, the resolution complexity of the random CSP $\mathcal{B}_{n,m}^{2,k,t}$ is polynomial if the constraint density is larger than a certain value. This partly answers the open problem regarding the resolution complexity of random CSP inside the constraint tightness interval $k - 2 < t < 2^{k-1}$ ([113]). For $k = 3$, fixed $c > \frac{7}{3}$, and integer tightness t, our results actually show that $t = 2$ is the exact tightness threshold for the exponential resolution complexity since $\mathcal{B}_{n,m}^{2,k,t}$ with $t = 1$ is simply random 3-SAT and has an exponential resolution complexity [37].

The existence of upper bounds characterized by unsatisfiable 2-SAT subproblems raises concerns that $\mathcal{B}_{n,m}^{2,k,t}$ might be still flawed even if the tightness t is less than 2^{k-1}. However, this is not the case. Using a random hypergraph argument and the fact that a cycle of 2-clauses is satisfiable, it can be shown that for any fixed $t \leq 2^{k-1} - 1$, $\mathcal{B}_{n,m}^{2,k,t}$ does have a phase transition with a threshold lower bounded by $\frac{1}{k(k-1)}$.

Theorem 4.2.3. *For any fixed $t \leq 2^{k-1} - 1$ and $c = \frac{m}{n} < \frac{1}{k(k-1)}$, $\mathcal{B}_{n,m}^{2,k,t}$ is asymptotically almost surely satisfiable.*

Having established that $\mathcal{B}_{n,m}^{2,k,t}$ has a phase transition, it is obvious that the tightness t serves almost the same role as the parameter p in the (2+p)-SAT [9] to model the gradual changing from the first order transition to the second order transition. For each fixed constraint tightness $1 \leq t \leq 2^{k-1} - 1$, let $c_k(t)$ be the threshold of the constraint density of the satisfiability transition. When $t = 1$, we get the k-SAT model, and hence, $c_k(1)$ is exactly the k-SAT threshold. As the tightness t gradually increases, $c_k(t)$ decreases to a limit

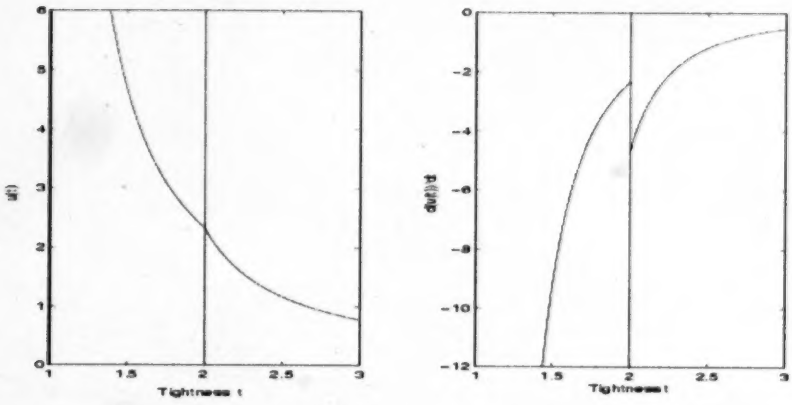

Figure 4.1: The upper bound $u(t)$ for the threshold $c_3(t)$ as a function of tightness t. Left figure: the function itself. Right figure the derivative of the function.

value larger than or equal to $\frac{1}{k(k-1)}$, continuously or discontinuously. Theorems 1 and 2 indicate that for random CSPs, it is possible to have different types of easy-hard complexity pattern if we can pick an appropriate relation between constraint tightness and constraint density. The property of the threshold as a function of the constraint tightness and constraint density deserves further investigation, and the behavior of the upper bounds in Theorem 4.2.1, as depicted in Figure 4.1, is suggestive. Some empirical results related to these issues will be reported in Chapter 5.

4.3 Proofs of the Results

4.3.1 Proof of Theorem 4.2.1

First, we give some definitions to be used to characterize CSP subproblems that imply unsatisfiable 2-SAT problems.

Definition 4.3.1 (k-Criss-Cross Loop). *Let $p > 0$ be an integer and $V = \{v_0, v_1, \ldots, v_{3p}\} \subset X = \{x_1, \ldots, x_n\}$ be a subset of variables. A k-criss-cross loop (k-cc-loop) $\mathcal{L}(V, E)$ is a k-uniform hypergraph on X whose hyperedges $E = \{E_1, \ldots, E_{3p+2}\}$ form two cycles $\mathcal{E}_1 = \{E_1, \ldots, E_{p+1}\}$ and $\mathcal{E}_2 = \{E_{p+2}, \ldots, E_{3p+2}\}$ such that*

1. $E_1 \cap E_{p+1} \cap E_{p+2} \cap E_{3p+2} = \{v_0\}$;
2. $E_i \cap E_{i+1} = \{v_i\}, \forall 1 \leq i \leq p$;
3. $E_i \cap E_{i+1} = \{v_{i-1}\}, \forall p+2 \leq i \leq 3p+1$; *and*

4. $\forall 1 \leq i \leq 3p+2, |E_i \setminus V| = k-2$, *and* $\{E_i \setminus V, 1 \leq i \leq 3p+2\}$ *are mutually disjoint.*

We call the variables in V the cyclic variables (or cyclic vertices) of the k-cc-loop. The variable v_0 is called the special variable of the k-cc-loop.

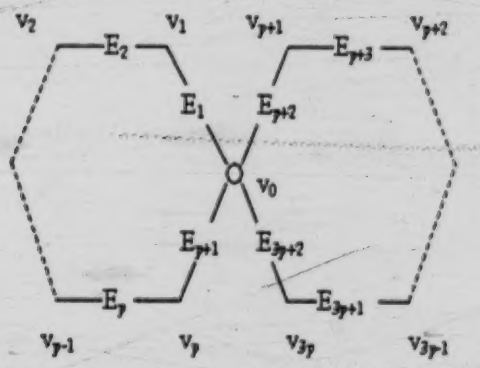

Figure 4.2: An illustration of a k-cc-loop. Only the cyclic variables $v_i, 1 \leq i \leq 3p$, are shown. Each hyper-edge E_i contains two cyclic variables from V and $(k-2)$ variables from $X \setminus V$.

In a k-cc-loop, there are exactly two cycles that touch at the special vertex v_0. This type of construct was first proposed by Franco in [60] and can be viewed as a generalization of the notion of simple cycles used in the study of the phase transition of random 2-SAT [36, 77]. The difference between the k-cc-loop defined in Definition 4.3.1 and those used in [36, 60, 77] is that the former is defined on variables while the latter are defined on literals.

Definition 4.3.2 (Reducible k-cc-loop). *Let $\mathcal{L}(V,E)$ be a k-cc-loop where $V = \{v_0, v_1, \ldots, v_{3p}\}$ and $E = \{E_1, \ldots, E_{3p+2}\}$. A sequence of constraints $\mathcal{C} = \{C_1, \ldots, C_{3p+2}\}$ is said to be a* **reducible k-cc-loop** *on $\mathcal{L}(V,E)$ if*

1. *Each C_i has E_i as its constraint scope;*
2. *Each C_i implies a 2-CNF clause defined on two cyclic variables in E_i such that the resulting set of 2-CNF clauses is of the form*

$$u_0 \vee u_1,\ \overline{u_1} \vee u_2,\ \overline{u_2} \vee u_3,\ \cdots,\ \overline{u_{p-1}} \vee u_p,\ \overline{u_p} \vee u_0;$$
$$\overline{u_0} \vee u_{p+1},\ \overline{u_{p+1}} \vee u_{p+2},\ \overline{u_{p+2}} \vee u_{p+3},\ \cdots,\ \overline{u_{3p-1}} \vee u_{3p},\ \overline{u_{3p}} \vee \overline{u_0},$$

where u_i is a literal of the variable v_i.

We call the above 2-CNF formula a contradictory bi-cycle on the k-cc-loop $\mathcal{L}(V,E)$.

In the following, we assume that $l = 3p + 2 = o(\sqrt{n})$.

Lemma 4.3.1. *Let $\mathcal{B}^{2,k,t}_{n,m}, c = \frac{m}{n}$, be a random CSP. Let $\mathcal{L}(V, E)$ be a k-cc-loop where $V = \{v_0, v_1, \ldots, v_{3p}\}$ is the sequence of cyclic variables and $E = \{E_1, \ldots, E_l\}$ is the sequence of hyperedges. The probability that $\mathcal{B}^{2,k,t}_{n,m}$ contains a reducible k-cc-loop on $\mathcal{L}(V, E)$ is*

$$\left(\frac{2rck!}{n^{k-1}}\right)^l \Theta(1),$$

where r is such that

1. *For $t = 2^{k-2} - 1 + \alpha$ with $0 < \alpha < 1$,*

$$r = \frac{1}{\binom{2^k}{2^{k-2}}}(1 + 2^{k-2}\alpha),$$

2. *For $t = 2^{k-2} + j + \alpha$ with $0 \leq \alpha < 1$ and $0 \leq j \leq 2^{k-1} - 2^{k-2} - 1$,*

$$r = \frac{\binom{2^{k-2}+j}{2^{k-2}}}{\binom{2^k}{2^{k-2}}}(1 + \alpha\frac{2^{k-2}}{j+1}).$$

Proof. Let $N = \binom{n}{k}$ be the number of possible hyperedges. Let

$$\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$$

be a sequence of constraints where each constraint C_i has the hyperedge E_i as its scope. Then, the probability that $\mathcal{B}^{2,k,t}_{n,m}, c = \frac{m}{n}$, contains the constraints $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ is

$$\frac{1}{\binom{N}{cn}}\binom{N-l}{cn-l} = \left(\frac{cn}{N}\right)^l \Theta(1) = \left(\frac{ck!}{n^{k-1}}\right)^l \Theta(1). \tag{4.3}$$

Let C be a constraint that has v_i and v_j as two of its scope variables. Given a literal u_i of the variable v_i and a literal u_j of the variable v_j, we calculate the probability that C implies the clause $u_i \vee u_j$. Here we give the details for the case of $t = 2^{k-2} + j + \alpha$ with $0 \leq \alpha < 1$ and $0 \leq j \leq 2^{k-1} - 2^{k-2} - 1$. The case of $t = 2^{k-2} - 1 + \alpha$ can be handled similarly.

Recall that a constraint contains a nogood set of size $t = 2^{k-2} + j$ with probability $1 - \alpha$ and of size $t = 2^{k-2} + j + 1$ with probability α. As we are dealing with constraints over Boolean variables, it is easy to see that the constraint C implies the clause $u_i \vee u_j$ if and only if the set of nogoods contains the set of 2^{k-2} binary vectors $(u_i, u_j, *)$ with $*$ being any binary vector in $\{0,1\}^{k-2}$. Therefore, the probability that C implies the clause $u_i \vee u_j$ is

$$\begin{aligned} r &= \frac{\binom{2^k - 2^{k-2}}{j}}{\binom{2^k}{2^{k-2}+j}}(1-\alpha) + \frac{\binom{2^k - 2^{k-2}}{j+1}}{\binom{2^k}{2^{k-2}+j+1}}\alpha \\ &= \frac{\binom{2^{k-2}+j}{2^{k-2}}}{\binom{2^k}{2^{k-2}}}(1 + \alpha\frac{2^{k-2}}{j+1}). \end{aligned} \tag{4.4}$$

As the constraint relations of the constraints are determined independently, the probability that the sequence of constraints $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ implies the 2-CNF contradictory bicycle defined by a literal sequence $(u_0, u_1, \cdots, u_{l-2})$ is r^l.

Since both of the positive and negative literals of the special variable v_0 have to appear in a 2-CNF contradictory bicycle, there are 2^{l-2} ways to select the literal sequences to form the contradictory bi-cycle. Since the constraint tightness t is less than 2^{k-1}, the events that the sequence of constraints $\mathcal{C}$ implies 2-CNF contradictory bi-cycles formed by different literal sequences are pair-wise disjoint. It follows that the probability for the sequence of constraints $\mathcal{C}$ to be a reducible k-cc-loop is

$$r^l 2^{l-2}. \tag{4.5}$$

The lemma is proved by combining (4.3), (4.4), and (4.5). □

Lemma 4.3.2. *For any $2^{k-2}-1 < t < 2^{k-1}$, the expected number of k-cc-loops on which the random CSP $B^{2,k,t}_{n,m}, c = \frac{m}{n}$, contains a reducible k-cc-loop is*

$$\frac{1}{4n}(2rck(k-1))^l \Theta(1)$$

where r is the same as in Lemma 4.3.1.

Proof. Let $V = \{v_0, v_1, \cdots, v_{3p}\}$ be a sequence of variables and $\mathcal{L}(V, E)$ be the k-cc-loop defined on V. From lemma 4.3.1, the probability that the CSP contains a reducible k-cc-loop on the k-cc-loop $\mathcal{L}(V, E)$ is

$$\left(\frac{2rck!}{n^{k-1}}\right)^l \Theta(1).$$

The total number of k-cc-loops is

$$\begin{aligned}
&\binom{n}{l-1}(l-1)! \prod_{i=0}^{l-1} \binom{n-l+1-(k-2)i}{k-2} \\
&= \binom{n}{l-1}(l-1)! \frac{1}{((k-2)!)^l} \frac{(n-l+1)!}{(n-l+1-l(k-2))!} \\
&= n^{l(k-1)} \frac{1}{((k-2)!)^l} \Theta(1)
\end{aligned}$$

where the term $\prod_{i=0}^{l-1} \binom{n-l+1-(k-2)i}{k-2}$ is the total number of ways to choose the variables for $E_i \setminus V$ for each hyperedge E_i in E. □

Proof of Theorem 4.2.1. Assume that $t \in (2^{k-2}-1, 2^{k-1})$ and $c = \frac{m}{n}$ satisfy one of the two conditions in Theorem 4.2.1. Let $p = \ln^2 n$ so that

$l = \Theta(1)\ln^2 n$. Let A_l be the number of k-cc-loops on which $\mathcal{B}_{n,m}^{2,k,t}$ contains a reducible k-cc-loop. To prove the theorem, it suffices to show that

$$\lim_n \Pr\{A_l > 0\} = 1. \tag{4.6}$$

Lemma 4.3.2 tells us that the expectation $\mathcal{E}[A_l]$ of A_l satisfies

$$\lim_{n\to\infty} \mathcal{E}[A_l] = \infty.$$

In order to use the second-moment method to establish (4.6), we claim that the variance $var(A_l)$ of A_l satisfies

$$var(A_l) = o(\mathcal{E}[A_l]^2).$$

For a k-cc-loop $\mathcal{L}(V, E)$ defined on V, let $I_{\mathcal{L}}$ be the indicator function of the event that $\mathcal{B}_{n,m}^{2,k,t}$ contains a reducible k-cc-loop on $\mathcal{L}(V, E)$. Then, $A_l = \sum_{\mathcal{L}} I_{\mathcal{L}}$ where the sum is over all the possible k-cc-loops. Given two k-cc-loops $\mathcal{L}$ and $\mathcal{M}$, we write $\mathcal{L} \sim \mathcal{M}$ if $\mathcal{L}$ and $\mathcal{M}$ share some hyperedges. Since $\mathcal{E}[I_{\mathcal{L}} I_{\mathcal{M}}] - \mathcal{E}[I_{\mathcal{L}}]\mathcal{E}[I_{\mathcal{M}}] = 0$ whenever the two k-cc-loops $\mathcal{L}$ and $\mathcal{M}$ do not share any hyperedges, we have

$$var(A_l) = \sum_{\mathcal{L}} var(I_{\mathcal{L}}) + \sum_{\mathcal{L}\sim\mathcal{M}} (\mathcal{E}[I_{\mathcal{L}} I_{\mathcal{M}}] - \mathcal{E}[I_{\mathcal{L}}]\mathcal{E}[I_{\mathcal{M}}]).$$

By the proof of lemma 4.3.2,

$$\mathcal{E}^2[A_l] = (\frac{1}{4n}(2rck(k-1))^l)^2 O(1).$$

Since

$$\sum_{\mathcal{L}} var(I_{\mathcal{L}}) = \sum_{\mathcal{L}} \mathcal{E}[I_{\mathcal{L}}](1 - \mathcal{E}[I_{\mathcal{L}}]) = o(\mathcal{E}^2[A_l]),$$

it is enough to show that

$$\sum_{\mathcal{L}\sim\mathcal{M}} \mathcal{E}[I_{\mathcal{L}} I_{\mathcal{M}}] = o(\mathcal{E}^2[A_l]). \tag{4.7}$$

Assume that $\mathcal{L}_1$ and $\mathcal{L}_2$ share q hyper-edges. Similar to the proof of lemma 4.3.1, we have

$$\begin{aligned} \mathcal{E}[I_{\mathcal{L}_1} | I_{\mathcal{L}_2}] &\le \frac{1}{\binom{N-l}{cn-l}} \binom{N-2l+q}{cn-2l+q} r^{l-q} 2^{l-q-2} && (4.8) \\ &= \left(\frac{2rck!}{n^{k-1}}\right)^{l-q} O(1) && (4.9) \end{aligned}$$

Therefore,

$$\mathcal{E}[I_{\mathcal{L}_1} I_{\mathcal{L}_2}] = \mathcal{E}[I_{\mathcal{L}_1} | I_{\mathcal{L}_2}]\mathcal{E}[I_{\mathcal{L}_2}] = \left(\frac{2rck!}{n^{k-1}}\right)^{2l-q} O(1). \tag{4.10}$$

To prove (4.7), we need to count the number of pairs of k-cc-loops sharing q hyper-edges. The counting technique is similar to those used in [60]. The following concepts about the cyclic variables in a k-cc-loop are required. Let $\mathcal{L}$ be a k-cc-loop and S be a set of hyper-edges in $\mathcal{L}$. We call a cyclic variable appearing in $\mathcal{L}$

1. *fixed* if it belongs to at least two hyper-edges in S;
2. *limited* if it belongs to one hyper-edge in S; and
3. *free* if it does not appear in any edges in S.

Write A_q for the total number of pairs of k-cc-loops sharing q hyperedges and $A_q(S)$ for the total number of pairs of k-cc-loops sharing a given set S of q hyperedges. We need to consider two different cases depending on the structure of the set of shared hyperedges S: (1) S is connected; and (2) S has $h \geq 2$ connected components. In each of the cases, we also need to distinguish how many of the 4 special hyperedges, i.e., the hyperedges that contain the special variable v_0, are shared.

Case 1: (The set of shared hyperedges S is connected) Let $q = |S|$. We consider three situations:

1. (*Each variable that appears in S is incident to at most two hyperedges of S .*) In this case, S is a hyper-path, and consequently any k-cc-loop that contains S will have $q-1$ fixed cyclic variables, 2 limited cyclic variables, and $(l-1-(q-1)-2)$ free cyclic variables. Therefore, the total number of pairs of k-cc-loops containing S is

$$\begin{aligned} A_q(S) &\leq \left(lk^2 n^{(l-1-(q-1)-2)} \binom{n}{k-2}^{l-q} \right)^2 \\ &= \frac{l^2 (n^{l-q-2} n^{(k-2)(l-q)})^2}{((k-2)!)^{2(l-q)}} O(1) \\ &= \frac{l^2}{n^4((k-2)!)^{2(l-q)}} n^{2(k-1)(l-q)} O(1). \end{aligned} \tag{4.11}$$

where the term l is for the number of possible positions of S in a k-cc-loop. As the number of possible hyper-paths with q hyper-edges is less than

$$H = \binom{n}{q-1}(q-1)!\binom{n}{k-1}^2\binom{n}{k-2}^{q-2} = nn^{(k-1)q}\frac{1}{((k-2)!)^q}O(1),$$

the total number of pairs of k-cc-loops sharing q hyperedges that form a hyper-path is less than

$$A_q(S) \cdot H \leq \frac{l^2}{n^3((k-2)!)^{2l-q}} n^{2(k-1)l} n^{-(k-1)q} O(1). \tag{4.12}$$

2. (*One variable v appears in three or four hyper-edges in S; The other variables are incident to at most two hyper-edges of S; And $q = |S| < p+2$*). In this case, S is a hyper-tree consisting of three or four hyper-path branches that join at the special variable v, as shown in Figure 4.3. If the degree of v in S is 3, then any k-cc-loop that contains S will have $q-2$ fixed cyclic variables, 3 limited cyclic variables, and $l-1-(q-2)-3$ free cyclic variables. Since the special variable v appears in S, the position of S in a k-cc-loop containing S is fixed. It follows that the number of pairs of k-cc-loops that share S is

$$\begin{aligned} A_q(S) &\leq \left(k^3 n^{l-1-(q-2)-3}\binom{n}{k-2}^{l-q}\right)^2 \\ &= \frac{1}{n^4((k-2)!)^{2(l-q)}} n^{2(k-1)(l-q)} O(1). \end{aligned} \tag{4.13}$$

The total number of such S, hyper-trees consisting of 3 hyper-path branches that join at special variables, is at most

$$\begin{aligned} H &= \binom{n}{q-2}(q-2)!\binom{n}{k-1}^3\binom{n}{k-2}^{q-3} \\ &= n^{q-2} n^{(k-2)(q-3)} n^{3(k-1)} O(1) \\ &= n n^{(k-1)q} \frac{1}{((k-2)!)^q} O(1). \end{aligned}$$

Then, the total number of pairs of k-cc-loops whose shared hyper-edges form a hyper-tree consisting of three hyper-path branches that join at the special variable is at most

$$\frac{1}{n^3((k-2)!)^{2l-q}} n^{2(k-1)l} n^{-(k-1)q} O(1). \tag{4.14}$$

Similar calculations show that the total number of pairs of k-cc-loops whose shared hyper-edges form a hyper-tree of four hyper-path branches that join at a special variable is less than (4.14).

3. (*One variable v appears in three or more hyper-edges in S; The other variables are incident to at most two hyper-edges of S; And $q = |S| \geq p+3$*). In this case, in addition to the cases where the shared hyperedges form a hyper-path or a hyper-tree consisting of hyper-path branches, we need to consider the situation where S forms a unicycle. If S forms a unicycle, then any k-cc-loop that contains S should have $q-1$ fixed cyclic variables and at least 1 limited cyclic variable. The total number of k-cc-loop pairs sharing a set S of hyper-edges that form a unicycle is at most

$$\frac{1}{n^2((k-2)!)^{2l-q}} n^{2(k-1)l} n^{-(k-1)q} O(1). \tag{4.15}$$

Figure 4.3: An illustration of a set of $q = 6$ shared hyper-edges that form a hyper-tree containing three hyper-path branches. The variable v appears in 3 hyper-edges. There are 4 fixed cyclic variables, 3 limited cyclic variables.

Case 2: (The set S of shared hyperedges form $h \geq 2$ connected components)

In this case, the total number of sets of shared hyperedges is more than that in Case 1. But this is compensated for by the decreasing number of of free cyclic variables. In the following, we discuss in detail the case where these h components are all hyper-paths. Other cases can be handled similarly. Let h_1 be the number of components in S that are isolated hyper-edges, h_2 be the number of components in S that contain 2 hyper-edges, and $h_3 = h - h_1 - h_2$ be the number of components in S that are hyper-paths of length greater than 2. There are $2h_1 + 2h_2 + 2h_3$ limited cyclic variables, $h_2 + ((q - h_1 - 2h_2) - h_3)$ fixed cyclic variables, and consequently $l - 1 - q - h$ free cyclic variables. Thus, the number of pairs of k-cc-loops that share S is at most

$$\begin{aligned} A_q(S) &= \left(l^h k^{2h} n^{l-1-q-h} \binom{n}{k-2}^{l-q} \right)^2 O(1) \\ &= \left(\frac{l^2 k^4}{n^2} \right)^h \frac{1}{n^2} n^{2(k-1)(l-q)} \frac{1}{((k-2)!)^{2(l-q)}} O(1). \end{aligned} \tag{4.16}$$

For the total number H of hyper-edge sets that form h hyper-path components, note that there are $(h_2 + ((q - h_1 - 2h_2) - h_3)) = q - h$ cyclic variables that are non-endpoints of the hyper-path components. Once these $q - h$ variables are fixed, there are at most n^{kh_1} ways to choose the single-edge components, $n^{2(k-1)h_2}$ ways to choose the hyper-edges for the hyper-path components whose length is 2, and $n^{2(k-1)h_3} \binom{n}{k-2}^{q-h_1-2h_2-2h_3}$ ways to choose the interior hyper-edges for the hyper-paths whose length is greater than 2. Therefore, the total number of hyper-edge sets of size q that form h hyper-path components is at most

$$\begin{aligned} & n^{q-h} n^{kh_1 + 2(k-1)(h_2+h_3) - k(h_1+2h_2+2h_3) + 2(h_1+2h_2+2h_3) + (k-2)q} \frac{1}{((k-2)!)^{q-h}} \\ = \; & n^{q-h} n^{2(h_1+h_2+h_3)+(k-2)q} \frac{1}{((k-2)!)^{q-h}} \\ = \; & n^h n^{(k-1)q} \frac{1}{((k-2)!)^{q-h}}. \end{aligned}$$

It follows that the total number of pairs of k-cc-loops sharing q hyper-edges that form h hyper-path components is at most

$$(l^2k^4(k-2)!)^h \frac{1}{n^{h+2}} n^{2(k-1)l} n^{-(k-1)q} \frac{1}{((k-2)!)^{2l-q}}.$$

Since $h \geq 2$ and $l = O(ln^2(n))$, we conclude that the total number of pairs of k-cc-loops sharing q hyper-edges that form h hyper-path components is less than Case 1, formula (4.12).

In summary, the number of pairs of k-cc-loops sharing a set of hyper-edges that form $h, h \geq 1$, components is dominated by the case of $h = 1$. Therefore, the total number of pairs of k-cc-loops sharing a set of q hyper-edges can be bounded as follows:

$$A_q \leq \begin{cases} \frac{l^2}{n^3((k-2)!)^{2l-q}} n^{2(k-1)l} n^{-(k-1)q} O(l), & \text{if } q \leq p+2 \\ \frac{1}{n^2((k-2)!)^{2l-q}} n^{2(k-1)l} n^{-(k-1)q} O(l), & \text{if } q > p+2, \end{cases} \tag{4.17}$$

where the term "$O(l)$" is a result of summing over all the ways in which the q hyper-edges are shared, i.e., the number of components and the structures of the components. Based on formulas (4.10) and (4.17), we have

$$\begin{aligned}
&\sum_{\mathcal{L}\sim\mathcal{M}} \mathcal{E}[I_{\mathcal{L}} I_{\mathcal{M}}] = \sum_{q=1}^{l} \left(\frac{2rck!}{n^{k-1}}\right)^{2l-q} A_q \\
&= \sum_{q=1}^{p+1} \left(\frac{2rck!}{n^{k-1}}\right)^{2l-q} \frac{l^2}{n^3((k-2)!)^{2l-q}} n^{2(k-1)l} n^{-(k-1)q} O(l) \\
&\qquad + \sum_{q=p+2}^{l} \left(\frac{2rck!}{n^{k-1}}\right)^{2l-q} \frac{1}{n^2((k-2)!)^{2l-q}} n^{2(k-1)l} n^{-(k-1)q} O(l) \\
&= O(l^3)\frac{1}{n^3}(2rck(k-1))^{2l} \sum_{q=1}^{p+1} (2rck(k-1))^{-q} \\
&\qquad + O(l)\frac{1}{n^2}(2rck(k-1))^{2l} \sum_{q=p+2}^{l} (2rck(k-1))^{-q} \\
&= \mathcal{E}^2(A_l)\frac{O(l^3)}{n} \sum_{q=1}^{p+1} (2rck(k-1))^{-q} + \mathcal{E}^2(A_l) O(l) \sum_{q=p+2}^{l} (2rck(k-1))^{-q} \\
&= \mathcal{E}^2(A_l) \left(\frac{O(l^3)}{n} + O(l)(2rck(k-1))^{-(p+2)}\right) \\
&= o(\mathcal{E}^2(A_l)),
\end{aligned}$$

where the last two equations are because of the assumptions that $2rck(k-1) > l$ and $l = 3p+2 = \Theta(ln^2 n)$. This establishes the formula (4.7) and thus, proves the theorem. □

4.3.2 Proof of Theorem 4.2.3

The proof of Theorem 4.2.3 is based on the concepts of and the results on hyper-trees and unicycles in random hypergraphs.

Consider the random k-homogenous constraint hypergraph $\mathcal{G}^k(n,m)$ of $\mathcal{B}^{2,k,t}_{n,m}$. From [97] (see Lemma 2.2.1), we have for any $c < \frac{1}{k(k-1)}$, $\mathcal{G}^k(n,m)$ **whp** consists of hyper-trees and unicyclic components. In this case, an instance of the random CSP is satisfiable if and only if the subproblems corresponding to the components of the constraint hypergraph are all satisfiable. A subproblem corresponding to a hyper-tree is satisfiable [115]. In the following, we prove that a subproblem corresponding to a unicyclic component is also satisfiable if the tightness of the constraint is less than 2^{k-1}. We break up the task into three lemmas.

Lemma 4.3.3. *For any uncyclic k-homogenous hypergraph $\mathcal{G}$ with the edge set $E = (E_1, \cdots, E_t)$, we have*

$$|E_i \cap E_j| \leq 2, \forall 1 \leq i, j \leq t.$$

Proof. Assume that $a = |E_i \cap E_j| > 2$ and let

$$G' = (V, E - \{E_i\}).$$

Notice that G' has at most $k-a+1$ connected components $\{G_1, \cdots, G_{k-a+1}\}$. Since a connected hypergraph has at least an excess of -1, we have

$$ex(\mathcal{G}) = ex(G_1) + \cdots + ex(G_{k-a+1}) + (k-1) \geq a - 2 > 0.$$

A contradiction to the unicyclicness of $\mathcal{G}$. □

Due to Lemma 4.3.3, we only need to consider unicycles in which edges have intersections of cardinality of at most 2.

Lemma 4.3.4. *Let $\mathcal{C}$ be a CSP such that*

1. *Its constraint graph $\mathcal{G}(V,E)$ is unicyclic ;*
2. *The tightness t is less than 2^{k-1}; and*
3. *There are a pair of hyper-edges E_i and E_j with $|E_i \cap E_j| = 2$.*

Then, $\mathcal{C}$ is satisfiable.

Proof. Let $G' = (V, E - \{E_i\})$. Since $|E_i \cap E_j| = 2$, there should be exactly $k-1$ connected components in G' such that (1) one of the components contains the intersection $E_i \cap E_j$, and each of the rest of the components contains exactly one vertex from $E_i - E_j$; and (2)each of the connected components has an excess of -1. Otherwise, $\mathcal{G}$ would have an excess larger than 0. The satisfiability of the CSP can be shown by first satisfying the constraint corresponding to the hyper-edge E_i and then satisfy other constraints. This is possible because for the tightness $t < 2^{k-1}$, there is always at least one assignment that satisfies E_i and E_j simultaneously. □

Now, we are in a position to deal with the situation where hyper-edges have an intersection of size at most 1.

Lemma 4.3.5. *Let $\mathcal{C}$ be a CSP such that*

1. *Its constraint graph $\mathcal{G}(V, E)$ is unicyclic ;*
2. *The tightness t is less than 2^{k-1}; and*
3. *For any pair of hyper-edges E_i and E_j, we have with $|E_i \cap E_j| \leq 1$.*

Then, $\mathcal{C}$ is satisfiable.

Proof. In this case, the constraint hypergraph $\mathcal{G}(V, E)$ contains one cycle $F = (F_1, \cdots, F_l)$ of the form

$$|F_i \cap F_{i+1}| = 1, 1 \leq i \leq l-1, |F_i \cap F_1| = 1.$$

and some additional hyper-tree branches attached to the cycle. If there is a partial assignment satisfying the constraints in the cycle, then we can always extend it to satisfy the constraints in the hyper-tree branches. To see there exists such a partial assignment, let $y_i = F_i \cap F_{i+1}$ and $y_n = F_n \cap F_1$. Consider the two possible assignments 0 and 1 to y_1. If we assign $y_1 = 0$ or 1, we can find assignments to $y_i, 2 \leq i \leq n-1$ to satisfy $F_1, \cdots, F_{n-1}$. Assume that y_n is forced to take the value a_0 for the assignment $y_1 = 0$ and a_1 for the assignment $y_1 = 1$. Since there are at most $2^{k-1} - 1$ restrictions to the variables in E_1, we know at least one of the pairs $(y_1 = 0, y_n = a_0)$ and $(y_1 = 1, y_n = a_1)$ will satisfy the constraint corresponding to F_1. This shows the existence of a partial assignment that satisfies the set of the constraints corresponding to the cycle $F = \{F_1, \cdots, F_l\}$. □

4.4 Discussions

After the results in this chapter first appeared in [72], Molloy and Salavatipour published their independent work in which the resolution complexity of $\mathcal{B}_{n,m}^{d,k,t}$ for the general case of $d \geq 2$ is investigated [116]. For the case of $d = 2$, their results lead to the same bounds as those presented in this chapter. More importantly, their work shows that for $t \in (2^{k-2}, 2^{k-1}]$, these bounds are in fact the exact thresholds, i.e., when the constraint density $\frac{m}{n}$ is below these bounds, $\mathcal{B}_{n,m}^{2,k,t}$ almost surely has an exponential resolution complexity. Their work also proves that $\mathcal{B}_{n,m}^{2,k,t}$ almost surely has an exponential resolution complexity for $t \in (k-1, 2^{k-2})$.

In summary, the resolution complexity of the model $\mathcal{B}_{n,m}^{2,k,t}$ has now been fully characterized.

Chapter 5

Consistency and Better Random CSP Models

5.1 Introduction

One of the significant problems with existing random CSP models with bounded domain size, including those that have been proposed recently, is that as a model parameter, the constraint tightness has to be very low in order to have non-trivial threshold behaviors and guaranteed hard instances at phase transitions. For random CSPs with increasing domain size, there is still a certain degree of restriction on the possible value of the constraint tightness.

As we have discussed in Section 3.3.1, except for a small range of the constraint tightness, all of the four classical random CSP models are trivially unsatisfiable with high probability due to the existence of flawed variables. For the case of binary CSPs, the constraint tightness has to be less than the domain size in order to avoid flawed variables. Furthermore, the results in Chapter 4 and in [116] show that even for a moderate constraint tightness, it is still possible for these classical models to have a polynomial complexity due to the appearance of embedded easy subproblems.

Several new models have been proposed to overcome the trivial unsatisfiability. Gent et al. [75] proposed the flawless random model for binary CSPs based on the notion of a *flawless conflict matrix*. Instances of the flawless random CSP model are guaranteed to be arc-consistent, and thus do not suffer asymptotically from the problem of flawed variables. Achlioptas et al. [8], proposed a nogood-based CSP model, the model E, and showed that it has non-trivial asymptotic behaviors. Random CSP models with a (slowly) increasing domain size have also been shown to be free from the problem of flawed variables and have interesting threshold behaviors [140, 129]. However, none of these models has specifically addressed the fundamental cause and requirement of a low constraint tightness in order to have a guaranteed exponential complexity. The flawless random CSP does have a true phase

transition of the solution probability at a high constraint tightness, but as we will show later, it still suffers from the embedded easy unsatisfiable sub-problems at a moderate constraint tightness. In CSP models with increasing domain size, there is still an obvious restriction on the possible values of the constraint tightness. In model E, it is impossible to have a high constraint tightness without making the constraint (hyper)graph very dense (see Lemma 3.2.2).

In this chapter, we study the possibility of designing non-trivial random CSP models that allow a much higher constraint tightness. This chapter is based on [73] published in 2004. We prove that there are strong connections between the resolution complexity of (randomly-generated) CSP instances and the *constraint consistency*, a notion that has been developed to improve the efficiency of CSP algorithms. These connections are somewhat surprising since almost all of the existing CSP algorithms exploit, in one way or another, the constraint consistency to improve their performance. We propose a scheme to generate consistent random CSP instances that can potentially have a high constraint tightness. Detailed experimental results are also reported to illustrate the sensitivity of instance hardness to the constraint tightness in classical CSP models and to show that instances generated by our model are indeed much harder at phase transitions than previous CSP models.

5.2 Consistency and Resolution Complexity of Random CSPs

Throughout this chapter, we consider binary CSPs defined on a domain $D = \{1, 2, \cdots, d\}$ such that $|D| = d$. Elements in D will usually be denoted by lower-case Greek letters, α, β, etc.

Constraint consistency is perhaps one of the most important concepts in the constraint programming literature [108]. Almost all of the (complete) CSP algorithms exploit, in one way or another, constraint consistency to improve their performance. Much effort has been spent to design efficient data structures and algorithms to achieve and maintain a certain level of constraint consistency before or during the backtracking search. For example, over seven algorithms with increasing efficiency have been proposed to maintain arc-consistency—the lowest level of non-trivial constraint consistency. For some special type of constraints, one has to solve a maximum bipartite graph matching problem to achieve constraint consistency.

Definition 5.2.1. *Let $(X = \{x_1, \cdots, x_n\}, C, D)$ be a binary CSP instance and $k \geq 1$ be an integer. We say that the instance is* ***k-consistent*** *if for any set of $(k-1)$ variables $X_{k-1} = \{x_{i_1}, \cdots, x_{i_{k-1}}\}$, any assignment $\{\delta_{i_1}, \cdots \delta_{i_{k-1}}\} \in D^{k-1}$ to X_{k-1} satisfying the sub-instance induced on X_{k-1}, and for any other*

variable $x_{i_k} \in X \setminus X_{k-1}$, *there is an assignment* δ_{i_k} *to* x_{i_k} *such that*

$$\{\delta_{i_1}, \cdots \delta_{i_{k-1}}, \delta_{i_k}\}$$

satisfies the sub-instance induced on $X_{k-1} \cup \{x_{i_k}\}$.

A CSP instance is called ***strongly k-consistent*** *if and only if it is* j*-consistent for each* $j \leq k$. *Of special interest are strong k-consistency for* $k = 1, 2, 3$, *also known as node-consistency, arc-consistency, and path-consistency.*

5.2.1 CNF Encoding of CSPs

Mitchell [113] developed a framework in which the notion of resolution complexity is generalized to CSPs and the resolution complexity of randomly-generated CSP instances can be studied. In this framework, the resolution complexity of a CSP instance is defined to be the resolution complexity of a natural CNF encoding which we give below.

Following [113], we call an expression of the form $x : \alpha$ a **literal** for a CSP. A literal $x : \alpha$ evaluates to TRUE at an assignment if the variable x is assigned the value α. With this notation, a nogood of a CSP can be viewed as a disjunction of the negations of a set of literals $x_i : \alpha_i, 1 \leq i \leq l$, and will be denoted by $\eta(x_1 : \alpha_1, \cdots, x_l : \alpha_l)$.

Definition 5.2.2 (CNF Encoding and Resolution Complexity of CSPs). *Given an instance* $\mathcal{T}$ *of a CSP on a set of variables* $\{x_1, \cdots, x_n\}$ *with the domain* $D = \{1, 2, \cdots, d\}$, *its CNF encoding* $\mathrm{CNF}(\mathcal{T})$ *is a CNF formula constructed as follows:*

1. *For every CSP variable* x_i, *there are* d *Boolean variables* $\{x_i : 1,\ x_i : 2, \ldots, x_i : d\}$ *each of which indicates whether or not* x_i *takes on the corresponding domain value.*

2. *For every CSP variable* x_i, *there is a clause*

   $$x_i : 1 \vee x_i : 2 \vee \cdots \vee x_i : d$$

   on the d *Boolean variables making sure that* x_i *takes at least one of the domain values;*

3. *For every nogood* $(\delta_1, \cdots, \delta_k) \in D^k$ *of each constraint* C *with the scope* $\mathrm{var}(C) = \{x_{i_1}, \cdots, x_{i_k}\}$, *there is a clause*

   $$\overline{x_{i_1} : \delta_1} \vee \cdots \vee \overline{x_{i_k} : \delta_k}$$

   to respect the nogood.

The resolution complexity of $\mathcal{T}$ *is defined to be the resolution complexity of its CNF encoding* $\mathrm{CNF}(\mathcal{T})$.

It is not hard to see that a CSP instance $\mathcal{T}$ is satisfiable if and only if its CNF encoding CNF($\mathcal{T}$) is satisfiable:

1. If $(\alpha_1, \cdots, \alpha_n) \in D^n$ is a satisfying assignment, then the CNF encoding CNF($\mathcal{T}$) can be satisfied by the truth assignment $x_i : \alpha_i = TRUE$ and $x_i : \beta = FALSE$ for every $\beta \neq \alpha_i$.

2. For any truth assignment that satisfies the CNF encoding CNF($\mathcal{T}$), the CSP instance can be satisfied by assigning x_i to any domain value α for which $x_i : \alpha = TRUE$.

In [113, 116], upper bounds on the constraint tightness t were established for the random CSPs to have an exponential resolution complexity. For random binary CSP $\mathcal{B}^{d,2,t}_{n,m}$, the bound is (1) $t < d - 1$; or (2) $t < d$ and $\frac{m}{n}$ is sufficiently small. For a moderate constraint tightness, as has been shown in Chapter 4 (see also [116]), it is still possible for these classical models to have an asymptotically polynomial complexity due to the existence of embedded easy subproblems. The primary reason for the existence of embedded easy subproblems is that with a moderate constraint tightness, constraints frequently imply constraints which force a pair of involved variables to take a single value-tuple.

Definition 5.2.3 (Forcers [116]). *A constraint C_f with* $\mathrm{var}(C_f) = \{x_1, x_2\}$ *is called an (α, β)-forcer if its nogood set is*

$$\mathrm{NG}(C_f) = \{\eta(x_1 : \alpha, x_2 : \gamma); \gamma \neq \beta\}.$$

We say that a constraint C contains an (α, β)-forcer C_f defined on the same set of variables as C if $\mathrm{NG}(C_f) \subseteq \mathrm{NG}(C)$.

Definition 5.2.4 (Forbidding cycles and flowers [116]). *An α-forbidding cycle for a variable x_0 is a set of constraints*

$$\{C_1(x_0, x_1), C_2(x_1, x_2), \ldots, C_{r-1}(x_{r-2}, x_{r-1}), C_r(x_{r-1}, x_0)\}$$

such that $C_1(x_0, x_1)$ is an (α, α_1)-forcer, $C_r(x_{r-1}, x_0)$ is an (α_{r-1}, α_r)-forcer $(\alpha_r \neq \alpha)$, and $C_i(x_{i-1}, x_i)$ is an (α_{i-1}, α_i)-forcer $(2 \leq i \leq r-1)$. We call x_0 the center variable of the α-forbidding cycle.

An r-flower $R = \{C_1, \cdots, C_d\}$ consists of d (the domain size) forbidding cycles each of which has the length r such that

1. *$C_i, 1 \leq i \leq d$, have the same center variable x;*

2. *each C_i is an α_i-forbidding cycle of the center variable x; and*

3. *these forbidding cycles do not share any other variables.*

5.2.2 Consistency and Resolution Complexity of Random CSPs

In the following, we show that it is not necessary to put restrictions on the constraint tightness in order to have a guaranteed exponential resolution complexity. Based on similar techniques as those used in the literature [113, 116, 21], we will show that if in $\mathcal{B}^{d,2,t}_{n,m}$, each constraint is chosen in such a way that the resulting instances are always strongly k-consistent ($k \geq 3$), then $\mathcal{B}^{d,2,t}_{n,m}$ has an exponential resolution complexity no matter how large the constraint tightness is.

Theorem 5.2.1. *Let $\mathcal{B}^{d,2,t}_{n,m}[SC]$ be a random CSP such that*

1. *its constraint graph is the standard random graph G(n, m); and*
2. *for each edge, the constraint relation is such that any instance of $\mathcal{B}^{d,2,t}_{n,m}[SC]$ is strongly k-consistent for any given $k \geq 3$.*

Then, the resolution complexity of $\mathcal{B}^{d,2,t}_{n,m}[SC]$ is **whp** *exponential.*

Proof. See Section 5.5. □

Using the tool developed in [116], the requirement of strong k-consistency for CSP instances to have an exponential resolution complexity can be further relaxed.

Definition 5.2.5. *A CSP instance is called weakly path-consistent if it is arc-consistent and satisfies the conditions of path-consistency for paths of length 3 or more.*

Theorem 5.2.2. *Let $\mathcal{B}^{d,2,t}_{n,m}[WC]$ be a random CSP such that*

1. *its constraint graph is the random graph G(n, m); and*
2. *for each edge, the constraint relation is such that any instance of $\mathcal{B}^{d,2,t}_{n,m}[WC]$ is weakly path-consistent and contains no forcer.*

Then, the resolution complexity of $\mathcal{B}^{d,2,t}_{n,m}[WC]$ is almost surely exponential.

Proof. See Section 5.5. □

5.3 Consistency Core and Harder Random CSP Model with High Constraint Tightness

Having established that random CSPs with a certain level of consistency have an exponential resolution complexity, the question remaining to be answered

is whether or not there are natural random CSP models that are guaranteed to be strongly k-consistent or weakly path-consistent. In fact, the CSP-encoding of the graph k-coloring problem is strongly k-consistent. Another example is the flawless random binary CSP proposed in [75] that is guaranteed to be arc-consistent, i.e., strongly 2-consistent. In this section, we discuss how to generate random CSPs with high tightness that are strongly 3-consistent or weakly path-consistent.

5.3.1 Flawless Random CSPs

Gent, et al [75] proposed the *flawless CSP model* to overcome the triviality of the classical random CSP models. A key observation is that the existence of flawed variables might be a direct result of the fact that classical random CSPs are not arc-consistent.

Definition 5.3.1 ($\mathcal{B}^{d,2,t}_{n,m}[1]$, Flawless Random Binary CSP). *In the flawless random binary CSP $\mathcal{B}^{d,2,t}_{n,m}[1]$, the constraint graph is defined in the same way as that in $\mathcal{B}^{d,2,t}_{n,m}$. For each constraint edge, the constraint relation is specified in two steps:*

1. *Choosing a random permutation π of $D = \{1, \cdots d\}$; and*
2. *Selecting a set of t value-tuples uniformly at random from $D \times D \setminus \{(i, \pi(i)), 1 \leq i \leq n)\}$ as the nogood set.*

For reasons that will become clear later in this section, we have used a suffix "[1]" in the symbol $\mathcal{B}^{d,2,t}_{n,m}[1]$ to indicate the fact that in the flawless random CSP, each value of each variable in any constraint is guaranteed to have one support value from the other variable. Consequently, a flawless random CSP is always arc-consistent and does not have flawed variables. However, even though the flawless random binary CSP $\mathcal{B}^{d,2,t}_{n,m}[1]$ does not suffer from the problem of trivial unsatisfiability, it can be shown that $\mathcal{B}^{d,2,t}_{n,m}[1]$ asymptotically has embedded easy subproblems for $t \geq d-1$ in the same way as the random binary CSP model $\mathcal{B}^{d,2,t}_{n,m}$.

Theorem 5.3.1. *For $t \geq d-1$, there is a constant $c^* > 0$ such that for any $\frac{m}{n} > c^*$,* **whp** *$\mathcal{B}^{d,2,t}_{n,m}[1]$ is unsatisfiable and can be solved in polynomial time.*

A detailed proof of Theorem 5.3.1 can be found in Section 5.5. The idea is to show that for $\frac{m}{n} > c^*$, the flawless random CSP $\mathcal{B}^{d,2,t}_{n,m}[1]$ contains **whp** an unsatisfiable subproblem called an r-flower. Furthermore, if a binary CSP instance contains an r-flower, then any path-consistency algorithm (see, e.g., [108]) will produce a new CSP instance in which the center variable of the r-flower has an empty domain.

5.3.2 Generalized Flawless Model and Consistency Core

We are now in a position to introduce our new CSP model, the generalized flawless random CSP, which enforces a higher level consistency and is guaranteed to have an exponential resolution complexity.

Definition 5.3.2 ($B^{d,2,t}_{n,m}[\mathcal{K}]$, Generalized Flawless Random Binary CSP). *In the generalized flawless random binary CSP $B^{d,2,t}_{n,m}[\mathcal{K}]$, $\mathcal{K}$ is a random bipartite graph with each part being the domain D of a variable. The constraint graph is defined in the same way as that in $B^{d,2,t}_{n,m}$. For each constraint edge, the constraint relation is specified as follows:*

1. *Generate the bipartite graph $\mathcal{K} = (D^2, E(\mathcal{K}))$ satisfying certain properties; and*
2. *Select a set of t value-tuples uniformly at random from $(D \times D) \setminus E(\mathcal{K})$ as the nogood set.*

The idea behind the generalized flawless random binary CSP is that by enforcing a subset of value-tuples (specified by the edges of the bipartite graph $\mathcal{K}$) to be always compatible, it is possible that the generated CSP instances will always satisfy a certain level of consistency. If we define $\mathcal{K}$ to be a 1-regular bipartite graph, then $B^{d,2,t}_{n,m}[\mathcal{K}]$ reduces to the flawless random binary CSP model $B^{d,2,t}_{n,m}[1]$.

The following result shows that a connected and l-regular bipartite graph $\mathcal{K}$ with sufficiently large l can be used to generate strongly 3-consistent random CSPs or weakly path-consistent random CSPs.

Theorem 5.3.2. *Let $\mathcal{K}$ be an l-regular connected random bipartite graph. Then, $B^{d,2,t}_{n,m}[\mathcal{K}]$ is always*

1. *strongly 3-consistent if and only if $l > \frac{d}{2}$; and*
2. *weakly path-consistent if and only if $l > \frac{d-1}{2}$.*

Proof. We only prove the case for the weak path-consistency and the case for the strong 3-consistency is similar.

Consider a path $x_1 - x_2 - x_3 - x_4$ and any assignment $x_1 = i$ and $x_4 = j$. There are l values of x_2 that are compatible to $x_1 = i$ and there are l values of x_3 that are compatible to $x_4 = j$. Since the bipartite graph is connected, there are at least $l + 1$ values of x_3 that are compatible to $x_1 = i$. Therefore if $l > (d-1)/2$, there must be a value of x_3 that is compatible to both $x_1 = i$ and $x_4 = j$.

To see the "only if" part, we will show that there is a connected bipartite graph $K(V,U)$ on two sets of vertices $V = \{v_1, v_2, \cdots, v_d\}$ and $U = \{v_1, u_2, \cdots, u_d\}$ such that the neighbors of the first l vertices in V are the first

$l+1$ vertices in U. First, we construct a complete bipartite graph on the vertex sets $\{v_1, v_2, \cdots, v_l\}$ and $\{u_1, u_2, \cdots, u_l\}$; second, we construct an l-regular connected bipartite graph on the vertex sets $\{v_{l+1}, \cdots, v_d\}$ and $\{u_{l+1}, \cdots, u_d\}$ such that (v_{l+1}, u_{l+1}) is an edge. We then replace the two edges (v_l, u_l) and (v_{l+1}, u_{l+1}) with two new edges (v_l, u_{l+1}) and (v_{l+1}, u_l). This gives the bipartite graph $K(V,U)$. The existence of such a bipartite graph $K(V,U)$ shows that when $l \leq \frac{d-1}{2}$, it is possible to have a constraint relation such that a constraint path of length 3 is not consistent. □

The generalized random CSP model $\mathcal{B}^{d,2,t}_{n,m}[\mathcal{K}]$ with a connected regular bipartite graph $\mathcal{K}$ allows a constraint tightness up to $\frac{(d+1)d}{2}$. The above theorem also indicates that this is the best possible constraint tightness when using an arbitrary connected bipartite graph $\mathcal{K}$. To achieve higher constraint tightness, we propose a recursive scheme to generate a bipartite graph $\mathcal{K}$ that is more efficient in its use of edges.

To facilitate the presentation, we call an l-regular connected bipartite graph $K(V,V)$ a *strong kernel* (or a *weak kernel*) on V if $l > \frac{|V|}{2}$ (respectively, $l > \frac{|V|-1}{2}$).

Definition 5.3.3 (Consistency Core). *Let $D_1 = D_2$ be the domains of two variables with $|D_1| = |D_2| = d$. The consistency core for the domains D_1 and D_2 is a bipartite graph $\mathcal{G}_{core}(D_1, D_2)$ on D_1 and D_2, and is defined recursively as follows.*

1. *If there are integers $s, c \geq 3$ such that $d = s \times c$, then*
   (a) *partition $D_i, i = 1, 2$, into s blocks $\{D_{ij}, 1 \leq j \leq s\}$ of equal size c;*
   (b) *build a strong (or weak) kernel $K(S,S)$ on the set $S = \{1, 2, \cdots, s\}$; and*
   (c) *let the edge set of $\mathcal{G}_{core}(D_1, D_2)$ be*

$$\cup\{\text{the edge set of } \mathcal{G}_{core}(D_{1i}, D_{2j}) : (i,j) \text{ are adjacent in the kernel } K(S,S)\}.$$

2. *Otherwise, $\mathcal{G}_{core}(D_1, D_2)$ is defined to be the strong (or weak) kernel on D_1 and D_2.*

It should be noted that in the above definition, if the domain size d (or $d/2$) is prime, the recursive steps will not happen and thus, the consistency core is simply a strong or weak kernel on D_1 and D_2. One way to make it work for prime numbers is to consider a fixed subset D' of the domain D such that $|D'|$ can be factorized. Once a consistency core has been built on D', a consistency core on D can be obtained by padding each element in $D \setminus D'$ to some block of the D'-partition and making it adjacent to every element in any other adjacent block.

Theorem 5.3.3. *If a consistency core is used for $\mathcal{K}$, then $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ is*

1. *strongly 3-consistent if and only if $l > \frac{s}{2}$; and*
2. *weakly path-consistent if and only if $l > \frac{s-1}{2}$.*

Proof. By induction on the domain size and using the previous theorem. □

Using the consistency core, we can define random CSP models with constraint tightness well above $\frac{(d+1)d}{2}$. For example, if the domain size d is 12, the random generalized random CSP model $\mathcal{B}^{d,2,t}_{n,m}[\mathcal{K}]$ with a consistency core $\mathcal{K}$ allow a constraint tightness up to $144 - 6 * 8 = 96$.

Generally, if $\{s_1, s_2, \ldots, s_q\}$ is the sequence of partition sizes used in the recursive steps when constructing a consistency core $\mathcal{K}$, then the highest achievable constraint tightness in the corresponding CSP is

$$T = d^2 - \prod_{k=1}^{q} s_k \lceil \frac{s_k}{2} \rceil.$$

We can therefore formulate the problem of finding an optimal sequence of partitions as the following optimization problem:

Given an integer $d > 0$, find a factorization

$$d = \prod_{k=1}^{q} s_k$$

such that $\prod_{k=1}^{q} s_k \lceil \frac{s_k}{2} \rceil$ is minimized.

In our generator, we have implemented a dynamic programming algorithm to generate such an optimal partition sequence.

Example 5.3.1. *Consider the consistency core $\mathcal{K}$ depicted in Figure 5.1 where the domain size is $|D| = 9$. The domain of a variable is partitioned into 3 blocks of size 3. The dashed lines are the edges of a strong kernel on the blocks. For each pair of blocks connected by a dashed line (e.g., the pair circled by the grey line), we build a 3 by 3 consistency core as depicted at the bottom of the figure. The edge set of the consistency core $\mathcal{K}$ consists of all the edges of all the 3 by 3 consistency core. In fact, there are in total 36 edges in $\mathcal{K}$. An instance of this CSP model can be viewed as a generalized 3-colorability problem.*

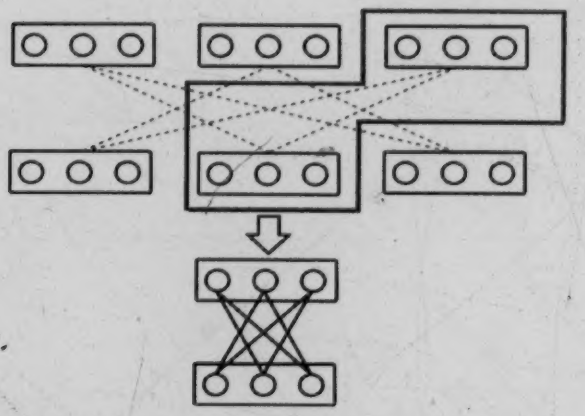

Figure 5.1: A special type of consistency core with the domain size 9

5.4 Experiments

In this section, we report results of two sets of experiments designed (1) to study the effect of an increase in the constraint tightness on the typical-case complexity; and (2) to compare the typical-case instance hardness between the classical random CSPs, the flawless random CSPs, and the generalized flawless random CSPs.

5.4.1 Effect of an Increase in Constraint Tightness

Upper bounds on the constraint tightness have been established for random CSPs to have an exponential resolution complexity for any constant constraint density $\frac{m}{n}$ [72, 116]. It was further shown in [116] that for the constraint tightness above the upper bound, the existence of forcers can be compensated for by sufficiently low constraint-to-variable ratio so that one can still have typical instances with exponential resolution complexity.

We have conducted the following experiments to gain further understanding of the effect of an increase in the constraint tightness (and hence an increase in the likelihood of the existence of a forcer in a constraint) on the typical-case hardness of random CSPs. The experiments also help understand the behavior of CSP models, such as the flawless CSP model, that only enforce arc-consistency (strong 2-consistency).

In the experiments, we start with a random 3-CNF formula whose clauses are treated as constraints. We then incrementally increase the tightness of each constraint by adding more clauses defined over the same set of variables. There are two reasons why we have based our experiments on random SAT models. First, the typical-case complexity of the random SAT model is well-understood and therefore, experiments based on the random SAT model will

enable us to have an objective comparison on the impact of an increase in the constraint tightness. Secondly, the complexity of Boolean-valued random CSPs obtained by increasing the tightness of the random 3-CNF formula has been characterized in great detail. We have a clear picture regarding the appearance of embedded easy subproblems in these Boolean-valued random CSPs [72].

Let $\mathcal{F}^3_{n,m}$ be a random 3-CNF formula with n variables and m clauses. We construct a new random 3-CNF formula $\mathcal{F}^{3,a}_{n,m}$ as follows:

1. $\mathcal{F}^{3,a}_{n,m}$ contains all the clauses in $\mathcal{F}^3_{n,m}$;
2. For each clause C in $\mathcal{F}^3_{n,m}$, we generate a random clause on the same set of variables of C, and add this new clause to $\mathcal{F}^{3,a}_{n,m}$ with probability a.

In fact, $\mathcal{F}^{3,a}_{n,m}$ is the random Boolean CSP model with a real-valued constraint tightness $1 + a$ and has been discussed in [72]. For $a > 0$, it is easy to see that $\mathcal{F}^{3,a}_{n,m}$ is always strongly 2-consistent, but is not 3-consistent asymptotically with probability 1.

Figure 5.3 shows the median of the number of branches used by the SAT solver zChaff on 100 instances of $\mathcal{F}^{3,a}_{n,m}$ with $n = 250$. Figure 5.2 shows the solution probability of the same model.

As expected, an increase in the tightness results in a shift of the location of the hardness peak toward smaller m/n. More significant is the magnitude of the decrease of the hardness as a result of a small increase in the constraint tightness. For example, we know [72] that the upper bounds on m/n for $\mathcal{F}^{3,a}_{n,m}$ to have an exponential resolution complexity are respectively 23.3 if $a = 0.1$ and 11.7 if $a = 0.2$. Since the constraint-to-variable ratios m/n considered in the experiment are well below these bounds above which embedded 2SAT subproblems appear with high probability, it seems that the impact of forcers on the instance hardness goes beyond simply producing embedded easy subproblems. As forcers can appear at a relatively low constraint tightness even in CSP models such as the flawless model, approaches that are solely based on restricting constraint tightness to generate interesting and typically hard instances cannot be as effective as has been previously believed.

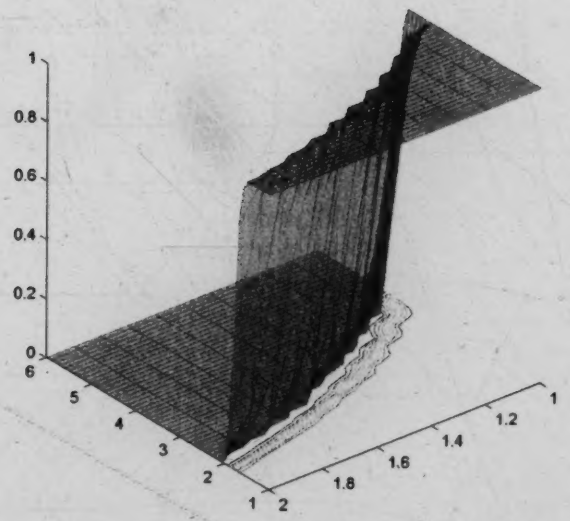

Figure 5.2: Thresholds for the solution probability in the model $\mathcal{F}_{n,m}^{3,a}$ with $n = 250$. The z-axis is the solution probability. The axis with the range 1—2 is for the parameter $1 + a$ and the axis with the range 1—6 is for the clause density m/n.

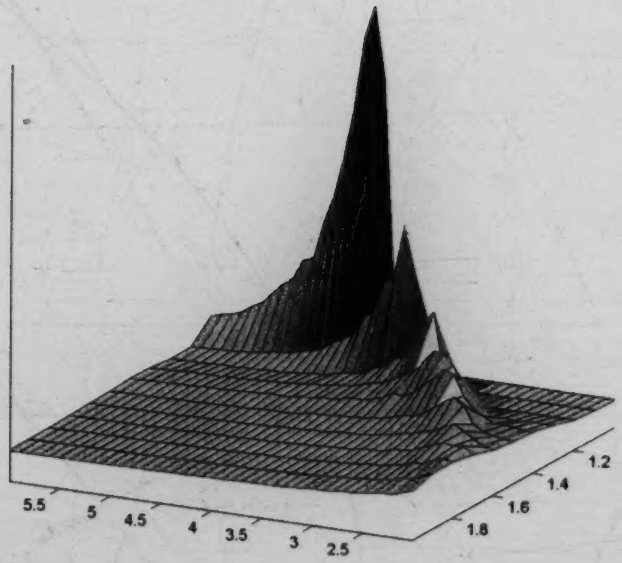

Figure 5.3: Effects of an increase in the constraint tightness on the instance hardness for $\mathcal{F}_{n,m}^{3,a}$ with $n = 250$. The z-axis is the median number of branches in log-scale. The axis with the range 1.2—1.8 is for the parameter $1 + a$ and the axis with the range 2.5—5.5 is for the clause density m/n.

5.4.2 Comparisons between Three Random CSP Models

This set of experiments is designed to investigate the effectiveness of the generalized flawless random CSP model. We generate random instances of the classical random models $\mathcal{B}^{d,t}_{n,m}$, the flawless random model $\mathcal{B}^{d,t}_{n,m}[1]$, and the generalized random model $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ with domain size $d = 4$. For $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$, we use a 2-regular connected bipartite graph as $\mathcal{K}$. These instances are then encoded as CNF formulas and solved by the SAT solvers zChaff [141] and Satz. It looks unnatural that we have tested random CSP instances by converting them to SAT instances and using a SAT solver. This is justified by the following considerations. First, all of the existing research on the resolution complexity of random CSPs has been carried out by studying the resolution complexity of a SAT encoding of CSPs as described in Section 5.2. We have used the same encoding in the experiments. Secondly, it has been shown that as far as the complexity of solving unsatisfiable CSP instances is concerned, many of the existing CSP algorithms can be efficiently simulated by the resolution system of the corresponding SAT encodings of the CSPs [112].

ZChaff

As Figure 5.4 shows, the threshold of the solution probability of the generalized random CSP model $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ is much sharper than those of $\mathcal{B}^{d,t}_{n,m}$ and $\mathcal{B}^{d,t}_{n,m}[1]$. More importantly, instances of $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ at the phase transition are much harder than those of $\mathcal{B}^{d,t}_{n,m}$ and $\mathcal{B}^{d,t}_{n,m}[1]$, as shown in Tables 5.1-5.3 where the median of the number of branches of zChaff for 100 instances of each of the three random CSP models is listed at different stages of the solubility phase transition: Table 5.1 is for the constraint density $\frac{m}{n}$ where the maximum median of the number of branches is observed; Table 5.2 is for the constraint density $\frac{m}{n}$ where the solubility probability is less than 0.1; and Table 5.3 is for the constraint density $\frac{m}{n}$ where the solubility probability is greater than 0.9.

It can be seen that while the classical random CSP model and flawless matrix CSP model have little difference, the proposed random CSP model $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ with $\mathcal{K}$ being a connected 2-regular bipartite graph is significantly harder in all of the cases except row 1 in Table 5.3. It is also interesting to notice that the most significant difference in the hardness among the three models is at the phase where instances of the random CSP models are almost always unsatisfiable. A plausible explanation for this phenomenon is that consistency is a property that may also help improve the efficiency of search algorithms in solving satisfiable instances.

The big differences between the proposed model and the other two models (the standard model and the flawless model) can be seen in Figure 5.5 where we plot the median number of branches of zChaff on 100 sample instances as a function of the constraints-variables ratio.

	Number of Branches		
(n,t)	$\mathcal{B}^{d,t}_{n,m}$	$\mathcal{B}^{d,t}_{n,m}[1]$	$\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$
$(100,6)$	235	228	391
$(300,6)$	2050	2017	5337
$(500,6)$	7655	8123	93649
$(300,8)$	843	1010	2785

Table 5.1: Maximum Median Number of Branches of zChaff on random instances of three random CSP models , over all $\frac{m}{n}$. Domain size $d = 4$ and $\mathcal{K}$ is 2-regular.

	Number of Branches		
(n,t)	$\mathcal{B}^{d,t}_{n,m}$	$\mathcal{B}^{d,t}_{n,m}[1]$	$\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$
$(100,6)$	128	178	312
$(300,6)$	840	1305	5311
$(500,6)$	2266	2553	52638
$(300,8)$	204	269	1118

Table 5.2: Median Number of Branches of zChaff on random instances of three random CSP models at the smallest $\frac{m}{n}$ where the solution probability is less than 0.1. Domain size $d = 4$ and $\mathcal{K}$ is 2-regular.

	Number of Branches		
(n,t)	$\mathcal{B}^{d,t}_{n,m}$	$\mathcal{B}^{d,t}_{n,m}[1]$	$\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$
$(100,6)$	221	204	169
$(300,6)$	2050	1572	2958
$(500,6)$	7655	6457	10632
$(300,8)$	843(0.67)	709	2785

Table 5.3: Median Number of Branches of zChaff on random instances of three random CSP models at the largest $\frac{m}{n}$ where the solution probability is greater than 0.9. Domain size $d = 4$ and $\mathcal{K}$ is 2-regular.

Model $B^{d,t}_{n,m}[\mathcal{K}]$					
m/n	1.2	1.6	2.2	2.4	2.6
Satz	170 (0.06)	139 (0.06)	68 (0.08)	47103 (44.47)	19230 (20.51)
ZChaff	1249 (0.01)	1839 (0.02)	8845 (0.44)	93649 (180.91)	18212 (10.85)
Model $B^{d,t}_{n,m}[1]$					
m/n	1.2	1.6	2.0	2.2	2.4
Satz	126 (0.05)	99 (0.05)	71 (0.07)	916 (0.76)	128 (0.17)
ZChaff	1384 (0.01)	2113 (0.02)	6457 (0.14)	5019 (0.43)	2123 (0.16)

Table 5.4: Median number of branches (median time in seconds) of ZChaff and Satz on two random CSP models with n = 500, d = 4, and t = 6. 100 instances for each parameter.

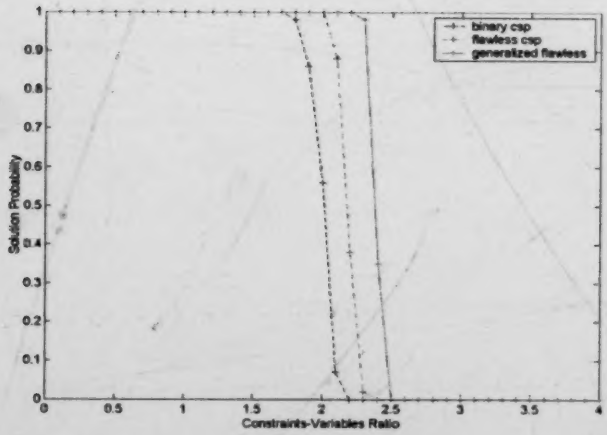

Figure 5.4: Solution probability thresholds for the three random CSP models with $n = 500, t = 6$. For the generalized flawless model, $\mathcal{K}$ is set to be 2-regular. The y-axis is solution probability and x-axis is the constraints-variables ratio m/n. Sample size for each data point is 100.

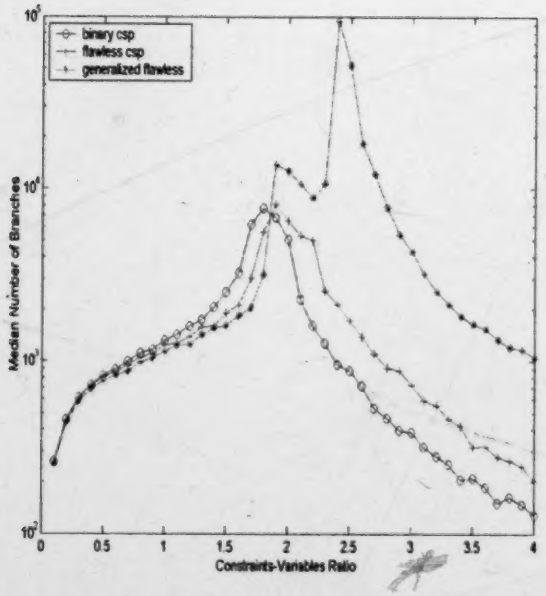

Figure 5.5: Hardness for the three random CSP models with $n = 500, t = 6$. For the generalized flawless model, $\mathcal{K}$ is set to be 2-regular. The y-axis is the median number of branches used by zChaff and x-axis is the constraints-variables ratio m/n. Sample size for each data point is 100.

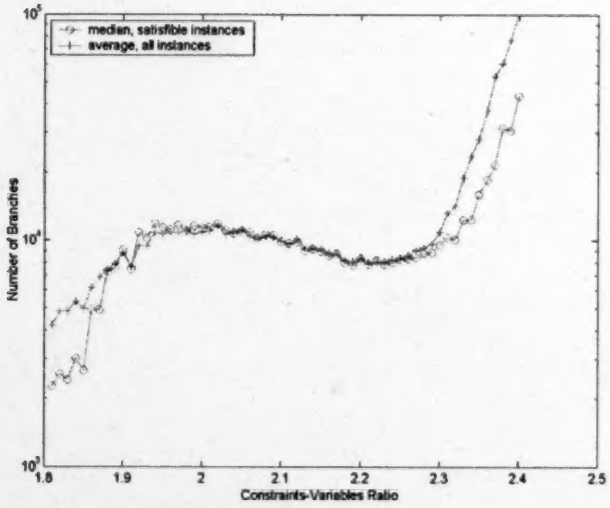

Figure 5.6: A closeup at the region $m/n = 1.8 - - - 2.5$ for the generalized flawless model with $n = 500, t = 6$ and $\mathcal{K}$ being 2-regular. Sample size for each data point is 200. Two curves are plotted. One is the median number of branches for satisfiable sample instances only, another is the average number of branches for all the sample instances.

Satz

Experimental comparison using another SAT solver, Satz, shows similar hardness patterns, and is summarized in Table 5.4. For both solvers, randomly-generated instances at the phase transition of model $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ are almost always much harder than those of the flawless model $\mathcal{B}^{d,t}_{n,m}[1]$ in terms of both the number of branches and the running time. It should be noticed that while the running time of the two solvers is comparable, the number of branches of Satz is much less. We believe this is a combined result of the different branching heuristics used by the two solvers as well as zChaff's excessive clause-caching overhead and memory usage.

Double peaks in instance hardness?

As has been depicted in Figure 5.5, in part of the satisfiable region of the generalized flawless model, the search cost, measured by the number of branches, has a small but not insignificant secondary peak. Initially, we had assumed that this is either a solver-specific behavior or a result of the finite sampling size (100 instances) we used when collecting the statistics. To our surprise, this secondary hardness peak persists in another set of experiments using zChaff where 200 instances were generated at each constraint density. See Figure 5.6 for a closeup view of the behavior of zChaff in this region. Experiments on both deterministic and randomized version of Satz also show a similar secondary peak. As we have measured the search cost by the median number of branches, the use of randomized version is not really necessary.

This leads us to speculate that the generalized flawless model might be the first model that shows solid evidence of the existence of double peaks in instance hardness. Recall that in the experiment, we used the generalized flawless model $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ with $d = 4$, $t = 6$ and a 2-regular connected consistency core. For smaller m/n, the consistency core in the generalized flawless model makes randomly-generated instances even easier. As m/n increases, perhaps right beyond 1.65—the threshold for the appearance of a 3-core [1] in the underlying random graph of the model, $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ should be such that any assignment that is compatible with the consistency core of each constraint cannot be a satisfying solution. Put another way, for any satisfying assignment $(\alpha_1, \cdots, \alpha_n)$, there must be at least one constraint C with a consistency core K_C such that the value-tuple (α_i, α_j) is not an edge in K_C. On the other hand, for any constraint in $\mathcal{B}^{d,t}_{n,m}[\mathcal{K}]$ with $t = 6$, the majority of the compatible value-tuples (8 out of 10) are specified by the consistency core. As a result, most search algorithms, if not all, will be misled to explore the consistency core part of the solution space in a certain degree to find out that any solution has to include at least one value-tuple outside of the consistency core. As m/n increases further, this becomes more and more obvious, resulting in a decreas-

[1] A 3-core of a graph is a maximum subgraph of the graph with minimum degree 3.

ing number of branches. Though there are several assumptions in the above speculation that need to be verified theoretically or empirically, we believe it provides a plausible explanation on the secondary peak.

5.5 Proof of the Theorems

In this section, we present more concepts related to the resolution complexity results stated in this paper and prove Theorems 5.3.1, 5.2.1, and 5.2.2.

5.5.1 Theorem 5.3.1

This subsection is devoted to Theorem 5.3.1. The following facts are straightforward to establish:

1. An r-flower consists of $s = d(r-1)+1 = dr-d+1$ variables and dr constraints;
2. The total number of r-flowers is
$$\binom{n}{s} s!(d-1)^d d^{d(r-1)}.$$
3. A constraint in the flawless CSP model contains an (α,β)-forcer only if the pair (α,β) is one of the pre-selected tuples in the flawless constraint matrix.

In the following, we assume that $r = o(\sqrt{n})$. The probability for a constraint to contain a forcer and the probability for the flawless random CSP to contain an r-flower are given in the following lemma.

Lemma 5.5.1. *Consider the flawless random CSP $B^{d,2,t}_{n,m}[1]$ and define $f_e = \frac{\binom{d^2-d-d+1}{t-d+1}}{\binom{d^2-d}{t}}$.*

1. *The probability that a given constraint $C(x_1,x_2)$ contains an (α,β)-forcer is*
$$\frac{1}{d} f_e. \tag{5.1}$$
2. *Let R be an r-flower and let $c = m/n$,*
$$\Pr\{\, R \text{ appears in } B^{d,t}_{n,m}[1] \,\} = \Theta(1)(2cf_e)^{dr}\frac{1}{n^{dr}}\frac{1}{d^{dr}}. \tag{5.2}$$

Proof. Equation (5.1) follows from the following two observations: (a)$\frac{1}{d}$ is the probability for (α, β) to be one of the pre-selected tuples in the flawless conflict matrix; and (b) f_e is the probability for the $d-1$ tuples, $(\alpha, \gamma), \gamma \neq \beta$, to be in the set of t restrictions selected uniformly at random from $d^2 - d$ tuples.

To calculate the probability that a given r-flower R appears in $\mathcal{B}_{n,m}^{d,2,t}[1]$, notice that the probability of selecting all the constraint edges in R is

$$\frac{\binom{N-dr}{cn-dr}}{\binom{N}{cn}} = \frac{cn(cn-1)\cdots(cn-dr+1)}{N(N-1)\cdots(N-dr+1)} = \Theta(1)\left(\frac{2c}{n}\right)^{dr}$$

where $N = \binom{n}{2}$. Since for each fixed choice of dr constraint edges in the r-flower, the probability for these constraints to contain the r-flower $(\frac{1}{d}f_e)^{dr}$, Equation (5.2) follows. □

Proof of Theorem 5.3.1. Let $c^* = \frac{d}{2f_e}$. We will show that if $c = \frac{m}{n} > c*$, then

$$\lim_n \Pr\{\, \mathcal{B}_{n,m}^{d,t}[1] \,\} \text{ contains an r-flower } \} = 1. \tag{5.3}$$

Let I_R be the indicator function of the event that the r-flower R appears in $\mathcal{B}_{n,m}^{d,2,t}[1]$ and let $X = \sum_R I_R$ where the sum is over all the possible r-flowers. Then, $\mathcal{B}_{n,m}^{d,t}[1]$ contains an r-flower if and only if $X > 0$.

By Lemma 5.5.1 and the fact that $s = dr - d + 1$, we have

$$\begin{aligned}
\mathcal{E}[X] &= \sum_R \mathcal{E}[I_R] \\
&= \Theta(1)\binom{n}{s} s!(d-1)^d d^{d(r-1)}(2cf_e)^{dr}\frac{1}{n^{dr}}\frac{1}{d^{dr}} \\
&= \Theta(1)n(n-1)\cdots(n-s+1)d^{dr}(2cf_e)^{dr}\frac{1}{n^{dr}}\frac{1}{d^{dr}} \\
&= \Theta(1)n^{1-d}(2cf_e)^{dr}.
\end{aligned}$$

Therefore, if $c > c^*$ and $r = \lambda \log n$ with λ sufficiently large, we have $\lim_n E[X] = \infty$.

If we can show that $\mathcal{E}[X^2] \leq \mathcal{E}^2[X](1 + o(1))$, then an application of the Chebyshev inequality will establish that $\lim_n \Pr\{\, X = 0 \,\} = 0$. To get an upper bound on $\mathcal{E}[X^2]$, we need a counting argument to upper bound the number of r-flowers sharing a given number of edges. This is done by considering how the shared edges form connected components [116, 72, 60]. Here, we follow the way that is used by Molloy and Salavatipour [116], from which we have

$$\begin{aligned}
\mathcal{E}[X^2] &= \sum_A \mathcal{E}[I_A^2] + \sum_A \sum_{B:B\cap A=\emptyset} \mathcal{E}[I_A I_B] + \sum_A I_A \left(\sum_{i=1}^{s} \sum_{j=1}^{i} N_{ij} (P_{ij})^{dr-i} \right) \\
&= \sum_A \mathcal{E}[I_A^2] + \sum_A \sum_{B:B\cap A=\emptyset} \mathcal{E}[I_A]\mathcal{E}[I_B] \\
&\qquad + \sum_A I_A \left(\sum_{i=1}^{s} \sum_{j=1}^{i} N_{ij} (P_{ij})^{dr-i} \right) \\
&\leq \mathcal{E}^2[X] + \sum_A I_A \left(\sum_{i=1}^{s} \sum_{j=1}^{i} N_{ij} (P_{ij})^{dr-i} \right) \qquad (5.4)
\end{aligned}$$

where (1) N_{ij} is the number of the r-flowers that share exactly i constraint edges with A and these i constraints forms j connected components in the constraint graph of A; and (2) $(P_{ij})^{dr-i}$ is the probability that conditional on I_A, the random CSP contains the $dr-i$ constraints of a specific r-flower as described in Lemma (5.5.1). In [116], N_{ij} is upper bounded by

$$\left((2+r^2)^d (dr^2)^{j-1}\right)^2 j! n^{s-i-j} d^{s-i-j+d-1},$$

where $((2+r^2)^d (dr^2)^{j-1})^2 j!$ upper bounds the number of ways to choose and arrange the j shared connected components for two r-flowers; n^{s-i-j} upper bounds the number of ways of choosing the remaining non-shared variables——the number of variables in each of the j shared connected component is at least one plus the number of edges in that shared component; and $d^{s-i-j+d-1}$ upper bounds the number of ways of choosing the forcing values in these non-sharing variables. The shared variables have to take the same forcing values as those in A due to the assumption that $t < d$ made in [116].

Since in our case $d-1 \leq t \leq d^2-d$, it is possible for shared variables to take different forcing values in different r-flowers. Thus, an upper bound for N_{ij} is

$$\left((2+r^2)^d (dr^2)^{j-1}\right)^2 j! n^{s-i-j} d^s.$$

But in our case, the probability corresponding to $(P_{ij})^{dr-i}$ is

$$\begin{aligned}
&\frac{\binom{N-dr-(dr-i)}{cn-i-(dr-i)}}{\binom{N-dr}{cn-i}} \left(\frac{1}{d} f_e\right)^{dr-i} = \Theta(1) \left(\frac{cn-i}{N-dr}\right)^{dr-i} \left(\frac{1}{d} f_e\right)^{dr-i} \\
&= \Theta(1) (2cf_e)^{dr-i} \frac{1}{n^{dr-i}} \frac{1}{d^{dr-i}}.
\end{aligned}$$

Therefore, with $c^* = \frac{d}{2f_e}$, we have

$$\begin{aligned}
&\sum_{i=1}^{s}\sum_{j=1}^{i} N_{ij}(2cf_e)^{dr-i}\frac{1}{n^{dr-i}}\frac{1}{d^{dr-i}} \\
\leq\ &\sum_{i=1}^{s}\left[(2+r^2)^{2d}r^{-4}n^{s-i}d^s(2cf_e)^{dr-i}\frac{1}{n^{dr-i}}\frac{1}{d^{dr-i}}\sum_{j=1}^{i}(\frac{d^2r^4j}{n})^j\right] \\
\leq\ &\sum_{i=1}^{s}O(r^{4d-4})n^{1-d}(2cf_e)^{dr}\frac{(2cf_e)^{-i}}{d^{-i}}O(\frac{r^4}{n}) \\
\leq\ &\mathcal{E}[X]O(r^{4d-4})O(\frac{r^4}{n})\sum_{i=1}^{s}(\frac{d}{2cf_e})^i \\
\leq\ &\mathcal{E}[X]O(\frac{r^{4d}}{n}),
\end{aligned}$$

where the last inequality is because $c > \frac{d}{2f_e}$. From this and formula (5.4), the proof is completed. □

Remark 5.5.1. *The relatively loose upper bound $c^* = \frac{d}{2f_e}$ in the above proof may be improved by a factor of d by making a further distinction among the r-flowers that share forcing values at different number of shared variables. But for the purpose of showing that the flawless random CSP also has potential embedded easy sub-problems, our upper bound for the constraint-variable ratio c is sufficient since the domain size d is a constant.*

5.5.2 Theorems 5.2.1 and 5.2.2

Let $\mathcal{T}$ be a CSP instance and let CNF($\mathcal{T}$) be the CNF encoding of $\mathcal{T}$. Mitchell [113] provided a framework within which one can investigate the resolution complexity of $\mathcal{T}$, i.e., the resolution complexity of the CNF formula CNF($\mathcal{T}$) that encodes $\mathcal{T}$, by working directly on the structural properties of $\mathcal{T}$. A sub-instance $\mathcal{J}$ of $\mathcal{T}$ is a CSP instance such that var($\mathcal{J}$) $\subset$ var($\mathcal{T}$) and $\mathcal{J}$ contains all the constraints of $\mathcal{T}$ whose scope variables are in var($\mathcal{J}$). The following crucial concepts make it possible to work directly on the structural properties of the CSP instance when investigating the resolution complexity of the encoding CNF formula.

Definition 5.5.1 (Implies. Defined in [113]). *For any assignment α to the variables in the CSP instance $\mathcal{T}$, we write $\hat{\alpha}$ for the truth assignment to the variables in* CNF($\mathcal{T}$) *that assigns to a variable x : a the value TRUE if and only if $\alpha(x) = a$.*

Let C be a clause over the variables of CNF($\mathcal{T}$). *We say that a sub-instance $\mathcal{J}$ of $\mathcal{T}$ implies C, denoted as $\mathcal{J} \models C$, if and only if for each assignment α satisfying $\mathcal{J}$, the corresponding $\hat{\alpha}$ satisfies C.*

Definition 5.5.2 (Clause Complexity [113]). *Let $\mathcal{T}$ be a CSP instance. For each clause C defined over the Boolean variables in* $\mathrm{var}(\mathrm{CNF}(\mathcal{T}))$*, define*

$$\mu(C,\mathcal{T}) = \min\{|\mathrm{var}(\mathcal{J})|; \mathcal{J} \text{ is a sub-instance and implies } C\}.$$

The following two concepts slightly generalize those used in [113, 116] and enable us to have a uniform treatment when establishing resolution complexity lower bounds.

Definition 5.5.3 (Boundary). *The boundary $\mathcal{B}(\mathcal{J})$ of a sub-instance $\mathcal{J}$ is defined to be the set of CSP variables such that $x \in \mathcal{B}(\mathcal{J})$ if and only if the following is true: If $\mathcal{J}$ minimally implies a clause C defined on some Boolean variables in* $\mathrm{var}(\mathrm{CNF}(\mathcal{T}))$*, then C contains at least one of the Boolean variables, $x : a, a \in D$, that encode the CSP variable x.*

Definition 5.5.4 (Sub-critical Expansion [113]). *Let $\mathcal{T}$ be a CSP instance. The sub-critical expansion of $\mathcal{T}$ is defined as*

$$e(\mathcal{T}) = \max_{0 \le s \le \mu(\emptyset,\mathcal{T})} \min_{s/2 \le |\mathrm{var}(\mathcal{J})| \le s} |\mathcal{B}(\mathcal{J})| \tag{5.5}$$

where the minimum is taken over all the sub-instances of $\mathcal{T}$ such that $s/2 \le |\mathrm{var}(\mathcal{J})| \le s$.

The following theorem relates the resolution complexity of the CNF encoding of a CSP instance to the sub-critical expansion of the CSP instance.

Theorem 5.5.1. *[113] For any CSP instance $\mathcal{T}$, we have*

$$w(\mathrm{CNF}(\mathcal{T}) \vdash \emptyset) \ge e(\mathcal{T}) \tag{5.6}$$

Proof. For any resolution refutation π of $\mathrm{CNF}(\mathcal{T})$ and $s \le \mu(\emptyset,\mathcal{T})$, Lemma 1 of [113] shows that π must contain a clause C with

$$s/2 \le \mu(C,\mathcal{T}) \le s.$$

Let $\mathcal{J}$ be a sub-instance such that $|\mathrm{var}(\mathcal{J})| = \mu(C,\mathcal{T})$ and $\mathcal{J}$ minimally implies C. Since $\mathcal{J}$ minimally implies C, according to the definition of the boundary, $w(C) \ge |\mathcal{B}(\mathcal{J})|$. (5.6) follows. □

To establish an asymptotically exponential lower bound on the resolution complexity of a random CSP $\mathcal{C}$, it is enough to show that there is a constant $\beta^* > 0$ that does not depend on n such that

$$\lim_n \Pr\{\, e(\mathcal{C}) \ge \beta^* n \,\} = 1. \tag{5.7}$$

For any $\alpha > 0$, let $\mathcal{A}_m(\alpha)$ be the event $\{\mu(\emptyset,\mathcal{C}) > \alpha n\}$ and $\mathcal{A}_s(\alpha,\beta^*)$ be the event

$$\left\{ \min_{\frac{\alpha n}{2} \le |\mathrm{var}(\mathcal{J})| \le \alpha n} \mathcal{B}(\mathcal{J}) \ge \beta^* n \right\}.$$

Notice that

$$\begin{aligned}\Pr\{e(\mathcal{C}) \geq \beta^* n\} &\geq \Pr\{\mathcal{A}_m(\alpha) \cap \mathcal{A}_s(\alpha, \beta^*)\} \\ &\geq 1 - \Pr\left\{\overline{\mathcal{A}_m(\alpha)}\right\} - \Pr\left\{\overline{\mathcal{A}_s(\alpha, \beta^*)}\right\}. \qquad (5.8)\end{aligned}$$

We only need to find appropriate α^* and β^* such that

$$\lim_n \Pr\left\{\overline{\mathcal{A}_m(\alpha^*)}\right\} = 0 \qquad (5.9)$$

and

$$\lim_n \Pr\left\{\overline{\mathcal{A}_s(\alpha^*, \beta^*)}\right\} = 0. \qquad (5.10)$$

The event $\mathcal{A}_m(\alpha^*)$ is about the size of minimally unsatisfiable sub-instances. For the event $\mathcal{A}_s(\alpha^*, \beta^*)$, a common practice is to identify a special subset of the boundary and show that the size of this subset is large. For different random CSP models and under different assumptions on the model parameters, there are different ways to achieve this. Following [21], we say a graph G is (r, q)-dense if there is a subset of r vertices that induces at least q edges of G.

Proof of Theorem 5.2.1. Recall that the constraint graph of $\mathcal{B}^{d,2,t}_{n,m}[SC]$ is the standard random graph $G(n, m)$. Since each instance of $\mathcal{B}^{d,2,t}_{n,m}[SC]$ is strongly k-consistent, variables in a minimal unsatisfiable sub-instance $\mathcal{J}$ with $|\mathrm{var}(\mathcal{J})| = r$ must have a vertex degree greater than or equal to k, and consequently, the constraint sub-graph $H(\mathcal{J})$ must contains at least $\frac{rk}{2}$ edges. Thus,

$$\begin{aligned}\Pr\left\{\overline{\mathcal{A}_m(\alpha^*)}\right\} &= \Pr\{\mu(\emptyset, \mathcal{B}^{d,t}_{n,m}[SC]) \leq \alpha^* n\} \\ &\leq \Pr\left\{\bigcup_{r=k+1}^{\alpha^* n} \{G(n, m) \text{ is } (r, rk/2)\text{-dense}\}\right\}.\end{aligned}$$

Let $\mathcal{B}^k(\mathcal{J})$ be the set of variables in $\mathrm{var}(\mathcal{J})$ whose vertex degrees are less than k. Again, since instances of $\mathcal{B}^{d,2,t}_{n,m}[SC]$ are always strongly k-consistent, we have $\mathcal{B}^k(\mathcal{J}) \subset \mathcal{B}(\mathcal{J})$ and thus, $|\mathcal{B}(\mathcal{J})| \geq |\mathcal{B}^k(\mathcal{J})|$. Therefore, the probability $\Pr\left\{\overline{\mathcal{A}_s(\alpha^*, \beta^*)}\right\}$ can be bounded as

$$\Pr\left\{\overline{\mathcal{A}_s(\alpha^*, \beta^*)}\right\} \leq \Pr\left\{\overline{\mathcal{A}^k_s(\alpha^*, \beta^*)}\right\}$$

where $\mathcal{A}^k_s(\alpha^*, \beta^*)$ is the event

$$\left\{\min_{\alpha^* n/2 \leq |\mathrm{var}(\mathcal{J})| \leq \alpha^* n} \mathcal{B}^k(\mathcal{J}) \geq \beta^* n\right\}.$$

Random graph arguments (see, e.g. [21]) show that there exist constants α^* and β^* such that $\Pr\left\{\overline{\mathcal{A}_m(\alpha^*)}\right\}$ and $\Pr\left\{\overline{\mathcal{A}^k_s(\alpha^*, \beta^*)}\right\}$ both tend to 0. Indeed,

let β^* be such that $\frac{(1-\beta^*)k}{2} > 1$, $c = \frac{m}{n}$, and $N = \frac{n(n-1)}{2}$. We have

$$\begin{aligned}
&\Pr\left\{\overline{\mathcal{A}_m(\alpha^*)}\right\} \le \Pr\left\{\bigcup_{r=k+1}^{\alpha^* n} \{G(n,m) \text{ is } (r, rk/2)\text{-dense }\}\right\} \\
&\le \sum_{r=k+1}^{\alpha^* n} \Pr\left\{G(n,m) \text{ is } (r, \frac{rk}{2})\text{-dense}\right\} \\
&\le \sum_{r=k+1}^{\alpha^* n} \binom{n}{r}\binom{\frac{r(r-1)}{2}}{\frac{rk}{2}}\binom{N-\frac{rk}{2}}{m-\frac{rk}{2}}\binom{N}{m}^{-1} \\
&\le \sum_{r=k+1}^{\alpha^* n} (\frac{en}{r})^r (\frac{e(r-1)}{k})^{\frac{rk}{2}} (\frac{2c}{n})^{\frac{rk}{2}} \\
&= \sum_{r=k+1}^{\alpha^* n} \left[\frac{en}{r}(\frac{2ec(r-1)}{kn})^{\frac{k}{2}}\right]^r \\
&= \sum_{r=k+1}^{\alpha^* n} \left[(\frac{k}{2})^{\frac{k}{2}} e^{\frac{k+2}{2}} c^{\frac{k}{2}} (\frac{r}{n})^{\frac{k-2}{2}}\right]^r \\
&\le \sum_{r=k+1}^{\lfloor \log n \rfloor} \left[(\frac{k}{2})^{\frac{k}{2}} e^{\frac{k+2}{2}} c^{\frac{k}{2}} (\frac{\log n}{n})^{\frac{k-2}{2}}\right] \\
&\qquad + \sum_{r=\lfloor \log n \rfloor}^{\alpha^* n} \left[(\frac{k}{2})^{\frac{k}{2}} e^{\frac{k+2}{2}} c^{\frac{k}{2}} (\alpha^*)^{\frac{k-2}{2}}\right]^{\log n} \qquad (5.11)
\end{aligned}$$

Similarly, we have for $\overline{\beta} = \frac{2\beta^*}{\alpha^*}$,

$$\begin{aligned}
&\Pr\left\{\overline{\mathcal{A}_s^k(\alpha^*, \beta^*)}\right\} \\
&= \Pr\left\{\bigcup_{r=\frac{\alpha^* n}{2}}^{\alpha^* n} \{\exists \text{ a size-r sub-instance } \mathcal{J} \text{ s.t. } |\mathcal{B}^k(\mathcal{J})| \le \beta^* n\}\right\} \\
&\le \Pr\left\{\bigcup_{r=\frac{\alpha^* n}{2}}^{\alpha^* n} \{G(n,m) \text{ is } (r, \frac{r(1-\overline{\beta})k}{2})\text{-dense}\}\right\} \\
&\le \sum_{r=\frac{\alpha^* n}{2}}^{\alpha^* n} \left[\left(\frac{2c}{(1-\overline{\beta})k}\right)^{\frac{(1-\overline{\beta})k}{2}} e^{\frac{(1-\overline{\beta})k+2}{2}} (\alpha^*)^{\frac{(1-\overline{\beta})k-2}{2}}\right]^r \qquad (5.12)
\end{aligned}$$

where the second inequality is because of the fact that for a sub-instance $\mathcal{J}$ with size r and $|\mathcal{B}^k(\mathcal{J})| \le \beta^* n$, its constraint graph contains at least $r - \beta^* n = r - \frac{\alpha^*}{2}\overline{\beta} n \ge r - \overline{\beta} r$ vertices whose degree is at least k.

There exist α^* and β^* be such that (1) $\frac{2\beta^*}{\alpha^*} < 1$; (2) $\frac{(1-\beta^*)k}{2} > 1$; and (3) the right hand side of formula (5.11) and the right hand side of formula (5.12) both tend to zero. This completes the proof of Theorem 5.3.1. □

We now prove Theorem 5.2.2. First from the definition of $\mathcal{B}^{d,t}_{n,m}[WC]$, we have the following

Lemma 5.5.2. *For the random CSP $\mathcal{B}^{d,t}_{n,m}[WC]$, we have*

1. *Every sub-instance whose constraint graph is a cycle is satisfiable;*
2. *For any path of length ≥ 3, any compatible assignments to the two variables at the ends of the path can be extended to assignments that satisfy the whole path.*

In an effort to establish exponential lower bounds on the resolution complexity for a classical random CSP models with a tightness higher than those in [113], Molloy and Salavatipour [116] introduced a collection of sub-instances, denoted here as $\mathcal{B}_M(\mathcal{J})$, and used its size to give a lower bound on the size of the boundary. For binary CSPs whose constraints are arc-consistent and contain no forcer, $\mathcal{B}_M(\mathcal{J})$ consists of two parts: $\mathcal{B}^1_M(\mathcal{J})$ and $\mathcal{B}^2_M(\mathcal{J})$, defined respectively as follows:

1. $\mathcal{B}^1_M(\mathcal{J})$ contains the set of single-edge sub-instances $\mathcal{X}$, i.e., $\mathrm{var}(\mathcal{X}) = 2$, such that at least one of the variables has a degree one vertex in the original constraint graph;
2. $\mathcal{B}^2_M(\mathcal{J})$ contains the set of sub-instances $\mathcal{X}$ whose induced constraint graph is a pendant path of length 4, i.e., a path of length 4 such that no vertex other than the endpoints has a vertex degree greater than 2 in the original constraint graph.

It can be shown that

Lemma 5.5.3 ([116]). *For any weakly path-consistent CSP sub-instance $\mathcal{J}$, we have*

$$|\mathcal{B}(\mathcal{J})| \geq |\mathcal{B}^1_M(\mathcal{J})| + \frac{|\mathcal{B}^2_M(\mathcal{J})|}{4}.$$

Proof. The variable with degree one in any sub-instance in $\mathcal{B}^1_M(\mathcal{J})$ has to be in $\mathcal{B}(\mathcal{J})$; At least one internal variable in any pendant path $\mathcal{B}^2_M(\mathcal{J})$ has to be in $\mathcal{B}(\mathcal{J})$. It is possible that several pendant paths of length 4 share a common internal variable that is in $\mathcal{B}(\mathcal{J})$, e.g., in a very long pendant path. But a variable can only appear in at most three pendant paths of length 4. □

With the above preparations, the proof provided for Theorem 1 of [116] readily applies to our case. To make this thesis self-contained, we give the proof below.

Proof of Theorem 5.2.2. By Lemma 5.5.2, any minimally unsatisfiable sub-instance $\mathcal{J}$ is such that (1) its constraint graph cannot be a single cycle; and (2) $\mathcal{B}_M(\mathcal{J})$ is empty since $|\mathcal{B}^1_M(\mathcal{J})| = 0$ and $|\mathcal{B}^2_M(\mathcal{J})| = 0$ for a minimally

unsatisfiable sub-instance. According to Lemma 11 of [116], the constraint graph of $\mathcal{J}$ has at least $(1+\frac{1}{12})\text{var}(\mathcal{J})$ edges. Therefore, due to the locally sparse property of random graphs (e.g., Lemma 10 in [116]), there is a constant $\alpha^* > 0$ such that formula (5.9) holds, i.e.,

$$\lim_n \Pr\left\{\overline{\mathcal{A}_m(\alpha^*)}\right\} = 0.$$

To establish formula (5.10), due to Lemma 5.5.3 we have

$$\Pr\{\mathcal{A}_s(\alpha^*,\beta^*)\} \geq \Pr\{\mathcal{A}_{s,M}(\alpha^*,\beta^*)\}$$

where $\mathcal{A}_{s,M}(\alpha^*,\beta^*)$ is the event

$$\left\{\min_{\alpha^* n/2 \leq |\text{var}(\mathcal{J})| \leq \alpha^* n} |\mathcal{B}_M(\mathcal{J})| \geq \beta^* n\right\}.$$

Now suppose on the contrary that there is a sub-instance $\mathcal{J}$ with $\alpha^* n/2 \leq |\text{var}(\mathcal{J})| \leq \alpha^*$ such that $|\mathcal{B}^1_M(\mathcal{J})| + |\mathcal{B}^2_M(\mathcal{J})| \leq \zeta n$. Then, from Lemmas 10 and 11 of [116], the constraint graph of $\mathcal{J}$ contains only cycle components—— Lemma 11 of [116] asserts that the edges-to-vertices ratio of the constraint graph of $\mathcal{J}$ has to be bigger than one. If we remove all the cycle components from the constraint graph of $\mathcal{J}$, the edges-to-vertices ratio of the remaining graph becomes even bigger. But this is impossible from Lemma 10 of [116] because the constraint graph of $\mathcal{J}$, and hence the remaining graph, has less than $\alpha^* n$ vertices.

It is well-known that **whp** a random graph has fewer than $\log n$ cycle components of length at most 4——for the random graph $G(m,n)$ with $m/n = c$ being constant, the number of cycle components with a fixed length has asymptotically Poisson distribution [30]. Thus, the number of variables that are in cycle components of length 4 is at most $4\log n$. Since any cycle component of length $l > 4$ contain l pendant paths of length 4, the total number of variables in cycle components of length greater than 4 is at most $|\mathcal{B}^2_M(\mathcal{J})| < \zeta n$. Therefore, we have $\text{var}(\mathcal{J}) < \zeta n + 4\log n < \alpha^* n/2 \leq \text{var}(\mathcal{J})$ for sufficiently small ζ, a contradiction.

We, therefore, conclude that there is a β^* such that w.h.p, for any sub-instance $\mathcal{J}$ with $\alpha^* n/2 \leq |\text{var}(\mathcal{J})| \leq \alpha^*$, $|\mathcal{B}_M(\mathcal{J})| \geq \beta^* n$, i.e., formula (5.10) holds. □

Chapter 6

Easy Random Problems Are Sometimes Hard

6.1 Introduction

It is well known that many NP-complete problems have tractable subclasses characterized by certain structural parameters. Treewidth is one such parameter and has drawn much attention in algorithmic graph theory [29, 103] and artificial intelligence [51].

In the study of the constraint satisfaction problem and the inference problems in Bayesian networks, there has been much effort in designing efficient algorithms that make best use of the property of having a bounded treewidth. The notion of tractable classes of CSPs parameterized by treewidth can be traced back to the work of [64] and since then, has remained an interesting topic [51, 52, 46, 81]. CSPs with bounded treewidth can be solved polynomially using dynamic programming techniques. Recently, treewidth has been shown to have a close connection with the complexity of some CSP proof systems [16].

In the study of model checking and planning, ordered binary decision diagram (OBDD) based propositional reasoning techniques have been widely used. OBDD-based satisfiability algorithms have been proposed and proved to have time complexity exponential in the treewidth of the underlying graph structures [121].

For Bayesian networks with a tree structure, the famous message-passing algorithm solves the inference problem in linear time [123]. For Bayesian networks with arbitrary structures, the most widely used method is the algorithm *join-tree* which transforms the original inference problem into the one on a *tree of subsets of variables*. The transformation is based on triangulation and tree-decomposition on the given network. As the size of the subsets in the tree-decomposition is directly related to the time and space complexities of the join-tree algorithm, there has been much work on finding the optimal

tree-decomposition, a problem that is also NP-hard [25, 92, 102]. Another recently proposed approach is to make sure that the Bayesain networks have a controlled treewidth when constructing/learning them from data[17, 96].

In the literature, there have been some attempts trying to relate the hardness of some of the above mentioned algorithms to the phase transition of the solution probability. In [111], Bayesian networks converted from randomly-generated CNF formulas were studied. In [121, 38], the complexity of OBDD-based satisfiability algorithms on randomly-generated CNF formulas were investigated. The initial intention of both of the studies seems to be a connection between the efficiency of these algorithms and the phase transition of the solution probability. However, in both cases, the authors concluded from their experimental results that the instance hardness for these algorithms has a dramatic change well below the phase transition threshold and keeps increasing beyond the threshold. The results presented in this chapter provide a theoretical justification for these experimental observations: Since the instance hardness for these dynamic programming algorithms is largely exponential in the treewidth of the problems, it is the phase transition of having a small treewidth that determines the behavior of these algorithms. As we will show in this chapter, the treewidth of the underlying structures of these problems has a phase transition from bounded-size to linear-size which occurs well below the solution probability threshold, and keeps increasing afterwards.

In Section 6.3, we establish an improved lower bound on the threshold for a random graph to have a linear treewidth. In Section 6.4, using similar analytical techniques, we show that the typical size of the treewidth of the underlying graph structures is also large in random CSPs, random Bayesian networks, and some other models in computational biology and evolutionary computation. These results, initially reported in our work [71, 69], indicate that several algorithms developed in the CSP and Bayesian network communities have a typically exponential behavior in the region of the problem space where randomly-generated instances can be solved easily by backtracking algorithms.

6.2 Notation and Definitions

The concepts of treewidth and tree-decomposition generalize that of a tree and characterize the degree to which a graph has a tree-like structure [103]. These concepts provide a viable way to characterize the degree of interaction in combinatorial structures and optimization problems. We discuss these concepts briefly and refer the interested reader to [29, 103, 34] for more details.

Treewidth can be defined in several equivalent ways. The one that is the easiest to state is via the *k-tree*.

Definition 6.2.1 (k-Tree[103]). *k-Trees are defined recursively as follows:*

1. *A clique with k+1 vertices is a k-tree;*

2. *Given a k-tree T_n with n vertices, a k-tree with n+1 vertices is constructed by adding to T_n a new vertex and connecting it to a k-clique of T_n.*

Definition 6.2.2 (Partial k-Trees and Treewidth[103]). *A graph is called a partial k-tree if it is a subgraph of a k-tree. The treewidth $tw(G)$ of a graph G is the minimum value k for which G is a partial k-tree.*

Treewidth has an equivalent definition based on tree-decomposition.

Definition 6.2.3 (Tree-decomposition [103]). *A tree-decomposition of a graph $G = (V, E)$ is a pair $\mathcal{D} = (\mathcal{S}, \mathcal{T})$ where $\mathcal{S} = \{S_i, i \in I\}$ is a collection of subsets of vertices of G and $\mathcal{T} = (I, F)$ is a tree with one node for each element in $\mathcal{S}$, such that*

1. $\bigcup_{i \in I} S_i = V$,

2. $\forall (v, w) \in E$ *there exists a subset* $S_i \in \mathcal{S}$ *such that both* v *and* w *are in* S_i, *and*

3. $\forall v \in V$, *the set of nodes* $\{i \in I;\ \ v \in S_i\}$ *forms a subtree of* $\mathcal{T}$.

The width of the tree-decomposition $\mathcal{D} = (\mathcal{S}, \mathcal{T})$ is $\max_{i \in I}(|S_i| - 1)$. The treewidth of a graph is the minimum width over all the tree-decompositions of the graph.

Treewidth has yet another equivalent definition based on the *minimum width* of a graph and the *vertex elimination* in a graph. It is also called the *induced width* in AI literature (see, for example, [50]).

Definition 6.2.4. *Let $G = (V, E)$ be a graph and $\pi = \{x_1, \cdots, x_n\}$ be an ordering of the vertices.*

1. *The* **width** $w(x, \pi)$ *of a vertex x under the ordering π is the number of its preceding neighbors. The* **width** $w(\pi)$ *of the ordering π is the maximum width of all the vertices under the ordering, i.e.,*

$$w(\pi) = \max_{x \in V} w(x, \pi).$$

2. *The* **π-induced graph** $G(\pi)$ *of G under the ordering π is obtained by processing the vertices recursively according to π from x_n to x_1. At each step i, all the neighbors of x_i that precede x_i according to π are made adjacent and then x_i is marked as processed. This process is called the vertex elimination.*

3. *The* **induced width** $w^*(G, \pi)$ *of G under the ordering π is the width of π in the π-induced graph $G(\pi)$ of G. The induced width $w^*(G)$ of G is the minimum induced width over all the vertex orderings.*

Given a graph G and a vertex ordering π, one can obtain a tree decomposition by (1) forming the induced graph $G(\pi)$; (2) identifying all the (maximum) cliques of $G(\pi)$; and (3) building a tree of this set of cliques in linear time that satisfies all the requirements of a tree decomposition.

In many applications, it is desirable to find a tree decomposition with a minimum width. This problem is NP-hard and has been an interesting topic in graph theory and artificial intelligence [29, 103, 34, 25].

6.3 Threshold of Linear Treewidth in Random Graphs

In [103], Kloks proved that **whp** a random graph $G(n,m)$ with $\frac{m}{n} > 1.18$ has a treewidth linear in n. Kloks commented that it was not known whether his lower bound 1.18 can be further improved and that the treewidth of a random graph $G(n,m)$ with $\frac{1}{2} < \frac{m}{n} < 1$ is unknown [103]. To my best knowledge, no further result has been obtained regarding the treewidth of $G(n,m)$ since Kloks' work.

In this section, we establish an improved lower bound on the threshold of having a linear treewidth. The improvement comes from two factors: (1) the use of a new combinatorial construct to make better use of the first moment method; and (2) the use of a random graph equivalent to $G(n,m)$ that makes it possible to have a more accurate estimation of some quantity.

Theorem 6.3.1. *For any $\frac{m}{n} = c > 1.081$, there is a constant $\delta > 0$ such that*

$$\lim_n \Pr\{\, tw(G(n,m)) > \delta n \,\} = 1. \tag{6.1}$$

We will be working on a random graph model $\overline{G}(n,m)$ that is slightly different from $G(n,m)$ in that the m edges are selected independently and uniformly with replacement. It turns out that as far as the property of having a linear treewidth is concerned, the two random graph models are equivalent. This is due to the following observations:

1. There are only $o(n)$ duplicated edges in $\overline{G}(n,m)$. In fact, let I_e be the indicator function of the event that the potential edge $e \in V^2$ is duplicated and write $I = \sum_{e \in V^2} I_e$. We have

$$\mathcal{E}[I_e] = \sum_{r \geq 2} \binom{m}{r} \frac{1}{N^r} (1 - \frac{1}{N})^{m-r} = O(\frac{1}{n^2}), \text{ where } N = \binom{n}{2}.$$

And thus, $\mathcal{E}[I] = O(1)$. On the other hand, we have for any pair of potential edges e_1 and e_2,

$$\mathcal{E}[I_{e_1} I_{e_2}] \leq \mathcal{E}[I_{e_1}]\mathcal{E}[I_{e_2}]$$

since I_{e_1} and I_{e_2} are negatively correlated. It follows that the variance of I is also $O(1)$, and therefore **whp** $I = o(n)$.

2. Due to the symmetry of the sampling space, a graph consisting of the first $m - o(n)$ non-duplicated edges of $\overline{G}(n, m)$ has the same distribution as $G(n, m - o(n))$.

3. For any graph G and its super-graph G' such that G' has $o(n)$ more edges than G, we have

$$tw(G') = tw(G) + o(n).$$

This is because adding one edge to a graph increases the treewidth of the graph at most by one.

Based on these observations, we will continue to use the notation $G(n, m)$ instead of $\overline{G}(n, m)$ throughout this section, but with the understanding that the m edges are selected independently and uniformly with replacement.

As the first step to prove theorem 6.3.1, we introduce the following concept which will be used to provide a necessary condition for a graph to have a treewidth of certain size:

Definition 6.3.1. *Let $G(V, E)$ be a graph with $|V| = n$. A partition $\mathbf{W} = (S, A, B)$ of V is said to be a rigid and balanced l-partition if the following conditions are satisfied:*

1. $|S| = l + 1$;
2. $\frac{1}{3}(n - l - 1) \leq |A|, |B| \leq \frac{2}{3}(n - l - 1)$; *and*
3. *S separates A and B, i.e., there are no edges between vertices of A and vertices of B; and*
4. *If $|B| > |A|$, then any vertex v in B is not isolated in B, i.e., there exists at least another vertex in B that is adjacent to v.*

A partition that satisfies the first three conditions in the above definition is called a balanced partition and was used by Kloks in his proof of the 1.18 lower bound. The rigid and balanced partition generalizes Kloks's balanced partition by requiring that any vertex in the larger subset of a partition cannot be moved to the other subset of the partition, and hence the word "rigid".

Lemma 6.3.1. *Any graph with a treewidth $l > 4$ must have a rigid and balanced $l-$partition.*

Proof. From [103], any graph with a treewidth $l > 4$ must have a partition, say $\mathbf{W} = (S, A, B)$, that satisfies the first three conditions in definition 6.3.1. If this partition does not satisfy the fourth condition, then we can move the vertices that are isolated in B one by one to A until either $|B| = |A|$ or there is no more isolated vertex in B. □

The following lemma gives an upper bound on the conditional probability for a partition $\mathbf{W} = (S, A, B)$ to be rigid given that the partition is balanced.

Lemma 6.3.2. *Let $G(n,m), c = \frac{m}{n}$, be a random graph and let $\mathbf{W} = (S, A, B)$ be a partition such that $|S| = l+1, |A| = a$, and $|B| = b$. Assume that $b = tn$ and $b > a$. Then for n sufficiently large,*

$$\Pr\{\mathbf{W} \text{ is rigid} \mid \mathbf{W} \text{ is balanced }\} \leq \left(\frac{1}{e}\right)^{r(t)n} \tag{6.2}$$

where

$$r(t) = \frac{t^2}{2c}\left(\frac{1}{e}\right)^{\frac{4ct}{1-2t(1-t)}}.$$

Proof. Conditional on that $\mathbf{W}$ is a balanced partition of $G(n,m)$, each of the m edges can only be selected from the set of edges

$$E_W = V^2 \setminus \{(u,v) : u \in A, v \in B\}.$$

Notice that

$$s \equiv |E_W| = \frac{n(n-1)}{2} - ba = \frac{n(n-1)}{2} - tn(n - tn - (l+1)).$$

Let I_v be the indicator function of the event that the vertex $v \in B$ is isolated in B and write $I = \sum_{v \in B} I_v$. Then, the random variable I is a function of the m outcomes when selecting the m edges of the random graph $G(n,m)$. For any two sets of outcome $(w_1, \cdots, w_m)$ and $(\overline{w}_1, \cdots, \overline{w}_m)$ that only differ at the i-th coordinate, i.e., the edges of two corresponding graphs are the same except for the i-th edge, we have

$$|I(w_1, \cdots, w_m) - I(\overline{w}_1, \cdots, \overline{w}_m)| \leq 2.$$

This is because changing one edge either increases or decreases the number of isolated vertices at most by two. Thus, applying McDiarmid's inequality (Lemma 2.1.5) gives us

$$\begin{aligned}
&\Pr\{\mathbf{W} \text{ is rigid} \mid \mathbf{W} \text{ is balanced }\} = \Pr\{I = 0 \mid \mathbf{W} \text{ is balanced }\} \\
&\leq \Pr\{I - \mathcal{E}[I] \leq -\mathcal{E}[I]\} \\
&\leq \left(\frac{1}{e}\right)^{\frac{2\mathcal{E}^2[I]}{4cn}}.
\end{aligned}$$

By the definition of the random variable I, the term $\mathcal{E}[I]$ is

$$b\left(1 - \frac{b-1}{s}\right)^{cn} = tn\left(1 - \frac{tn-1}{n(n-1)/2 - tn(n-tn-l-1)}\right)^{cn}$$

Formula (6.2) follows. □

We need two more lemmas on the behavior of some functions that will be used in the proof of Theorem 6.3.1.

Lemma 6.3.3. *For any $c > 1$, the function $r(t)$ in Lemma 6.3.2 is monotone-decreasing on $[\frac{1}{2}, \frac{2}{3}]$.*

Proof. Taking the derivative of the function

$$\log(r(t)) = 2\log(t) - \frac{4ct}{1 - 2t + 2t^2},$$

we have

$$\frac{1}{r(t)} r'(t) = \frac{2(1 - 2t + 2t^2)^2 - 4c(t - 2t^2 + 2t^3) - 4c(-2t^2 + 4t^3)}{t(1 - 2t + 2t^2)^2}.$$

Now consider the numerator of the right-hand-side of the above, i.e., the function

$$h(t) = 2(1 - 2t + 2t^2)^2 - 4c(t - 2t^2 + 2t^3) - 4c(-2t^2 + 4t^3).$$

The monotonicity of the function $r(t)$ can be established if we can show that $h(t) \leq 0, \forall t \in [\frac{1}{2}, \frac{2}{3}]$. Since we have $h(\frac{1}{2}) = \frac{1}{2} - c < 0$ and $h(\frac{2}{3}) = \frac{50}{81} - \frac{144}{81}c < 0$, it is enough to show that $h(t)$ itself is monotone. The first and second derivatives of the function $h(t)$ are respectively

$$h'(t) = 4(-2 + 8t - 12t^2 + 8t^3) - 4c(1 - 8t + 18t^2)$$

and

$$h''(t) = 4[(8 - 24t + 24t^2) - c(-8 + 36t)].$$

Notice that as a quadratic polynomial, $h''(t) = 4(24t^2 - (24 + 36c)t + 8(1 + c))$ can be shown to be always less than 0 for any $t \in [\frac{1}{2}, \frac{2}{3}]$. Since $h'(\frac{1}{2}) = -4c(1 + \frac{1}{2}) < 0$, it follows that $h'(t) < 0, \forall t \in [\frac{1}{2}, \frac{2}{3}]$. Therefore $h(t)$ is monotone as required. □

Lemma 6.3.4. *Let $g(t)$ be a function defined as*

$$g(t) = \frac{(1 - 2t + 2t^2 + 2\delta t)^c}{t^t (1 - t)^{1-t}} \tag{6.3}$$

where $c > 1$ and $\delta > 0$ are constants. Then, for small enough δ, $g(t)$ is monotone-increasing on $[\frac{1}{2}, \frac{2}{3}]$.

Proof. Consider the function $h(t) = \log g(t)$

$$h(t) = c\log(1 - 2t + 2t^2 + 2\delta t) - t\log t - (1 - t)\log(1 - t).$$

We have

$$h'(t) = c\frac{-2 + 4t + 2\delta}{1 - 2t + 2t^2 + 2\delta t} - logt + log(1 - t)$$

and $h'(\frac{1}{2}) \geq 0$. The second-order derivative of $h(t)$ is

$$h''(t) = \frac{c}{(1-2t+2t^2+2\delta t)^2 t(1-t)} \times z(t,\delta)$$

where

$$z(t,\delta) = 4(1-2t+2t^2+2\delta t)(1-t)t - (4t-2+2\delta)^2(1-t)t - (1-2t+2t^2+2\delta t)^2.$$

First, assume that $\delta = 0$. On the interval $[\frac{1}{2}, \frac{2}{3}]$, we have

$$(4t-2+2\delta)^2 \leq (4 \times \frac{2}{3} - 2)^2 = \frac{4}{9},$$

$$\frac{2}{9} \leq t(1-t) \leq \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$$

and

$$\frac{1}{2} \leq (1-2t+2t^2+2\delta t)^2 \leq (1-2\times\frac{2}{3}+2\times(\frac{2}{3})^2)^2 = \frac{5}{9}.$$

It follows that

$$z(t,\delta=0) \geq 4 \times \frac{1}{2}\frac{2}{9} - \frac{1}{9} - (\frac{5}{9})^2 = \frac{2}{81} > 0.$$

Since the family of functions $z(t,\delta), \delta > 0$ are uniformly continuous on $[\frac{1}{2}, \frac{2}{3}]$, we have that for small enough δ, $z(t,\delta) > 0$. Therefore, the second-order derivative $h''(t)$ is always larger than zero. And so is $h'(t)$ (recall that $h'(\frac{1}{2}) > 0$). It follows that $h(t)$ is monotone-increasing, and so is $g(t)$. □

Proof of Theorem 6.3.1

Proof. Let $\mathbf{W} = (S, A, B)$ be a partition of the vertices of $G(n,m)$ such that $|S| = l+1 = \beta n, |B| \geq |A|, |B| = b = tn$, with $\frac{1}{2} \leq t \leq \frac{2}{3}$. Let $I_{\mathbf{W}}$ be the indicator function of the event that $\mathbf{W}$ is a rigid and balanced l-partition of $G(n,m)$. We have

$$\begin{aligned} \mathcal{E}[I_{\mathbf{W}}] &= \Pr\{\mathbf{W} \text{ is rigid and balanced}\} \\ &= \Pr\{\mathbf{W} \text{ is balanced}\} \Pr\{\mathbf{W} \text{ is rigid} \mid \mathbf{W} \text{ is balanced}\} \quad (6.4) \end{aligned}$$

From Lemma 6.3.2, we know that

$$\Pr\{\mathbf{W} \text{ is rigid} \mid \mathbf{W} \text{ is balanced}\} \leq \left(\frac{1}{e}\right)^{r(t)n}$$

By the definition of a balanced partition,

$$\begin{aligned} \Pr\{\mathbf{W} \text{ is balanced}\} &= \left(1 - \frac{tn(n-tn-\beta n)}{n(n-1)/2}\right)^{cn} \\ &= \left[1 - 2t + 2t^2 + 2t\beta + O(1/n)\right]^{cn}. \quad (6.5) \end{aligned}$$

This is because in order for $\mathbf{W}$ to be a balanced partition of $G(n,m)$, each of the m independent trials can only select an edge from the set of vertex pairs $V^2 \setminus \{(u,v) : u \in A, v \in B\}$. Write

$$\phi_1(t) = \left[1 - 2t + 2t^2 + 2t\beta + O(1/n)\right]^c,$$

$$\phi_2(t) = \left[\left(\frac{1}{e}\right)^{\frac{1}{c}r(t)}\right]^c$$

and

$$\phi(t) = \phi_1(t)\phi_2(t)$$

so that we have

$$\mathcal{E}[I_{\mathbf{W}}] = [\phi(t)]^n.$$

Let $I = \sum_{\mathbf{W}} I_{\mathbf{W}}$ be the number of rigid and balanced l-partitions of the random graph $G(n,m)$ where the sum is taken over all such possible partitions. For a partition (S,A,B), there are $\binom{n}{\beta n}$ ways to choose the vertex set S with $|S| = \beta n$. For a fixed vertex set S, there are $\binom{n-\beta n}{b}$ ways ($\frac{1}{2}n \le b \le \frac{2}{3}n$) to choose the pair (A,B) such that one of them has the size b. Therefore,

$$\begin{aligned}
\mathcal{E}[I] &= \sum_{\mathbf{W}} \mathcal{E}[I_{\mathbf{W}}] \\
&\le \binom{n}{\beta n} \sum_{\frac{1}{2}n \le b \le \frac{2}{3}n} \binom{n-\beta n}{b} [\phi(\frac{b}{n})]^n \\
&\le \binom{n}{\beta n} \sum_{\frac{1}{2}n \le b \le \frac{2}{3}n} \binom{n}{b} [\phi(\frac{b}{n})]^n.
\end{aligned}$$

By Stirling's formula, we have for n large enough

$$\mathcal{E}[I] \le \left(\frac{1}{\beta^\beta (1-\beta)^{1-\beta}}\right)^n \sum_{\frac{1}{2}n \le b \le \frac{2}{3}n} \left(\frac{\phi_1(\frac{b}{n})\phi_2(\frac{b}{n})}{\frac{b}{n}^{\frac{b}{n}}(1-\frac{b}{n})^{1-\frac{b}{n}}}\right)^n$$

By Lemma 6.3.3,

$$\phi_2(\frac{b}{n}) \le \phi_2(\frac{2}{3}) = \left[\left(\frac{1}{e}\right)^{\frac{2}{9c^2}(\frac{1}{e})^{4.8c}}\right]^c$$

By Lemma 6.3.4,

$$\frac{\phi_1(\frac{b}{n})}{\frac{b}{n}^{\frac{b}{n}}(1-\frac{b}{n})^{1-\frac{b}{n}}} \le \frac{\phi_1(\frac{2}{3})}{(\frac{2}{3})^{\frac{2}{3}}(\frac{1}{3})^{\frac{1}{3}}} = \frac{(\frac{5}{9} + \frac{4}{3}\beta)^c}{(\frac{2}{3})^{\frac{2}{3}}(\frac{1}{3})^{\frac{1}{3}}}.$$

Therefore,

$$\mathcal{E}[I] \leq O(n) \left(\frac{1}{\beta^{\beta}(1-\beta)^{1-\beta}} \right)^n \left(\frac{[(\frac{5}{9} + \frac{4}{3}\beta)\left(\frac{1}{e}\right)^{\frac{2}{9c^2 e^{4.8c}}}]^c}{(\frac{2}{3})^{\frac{2}{3}}(\frac{1}{3})^{\frac{1}{3}}} \right)^n.$$

From the above, it can be shown that for sufficiently small β and $c > 1.081$,

$$\mathcal{E}[I] \leq O(n)\gamma^n$$

with $0 < \gamma < 1$. The theorem then follows from Markov's inequality and Lemma 6.3.1:

$$\lim_n \Pr\{ tw(G(n,m)) < \beta n \} \leq \lim_n \Pr\{ I > 0 \} \leq \lim_n \mathcal{E}[I] = 0.$$

□

Discussion: Why do I believe the threshold is less than one?

My conjecture that the threshold of having a linear treewidth is less than one (actually close to 1/2) is based on the size of the "giant" component in a random graph. Recall that Lemma 6.3.1 says that in order for a graph to have a treewidth $\leq l-1$, the graph must have a balanced partition $\mathbf{W} = (S, A, B)$ such that $|S| = l$ and $\frac{1}{3}(n-l) \leq |A|, |B| \leq \frac{2}{3}(n-l)$.

Consider the random graph $G(n,m)$ with $1/2 < \frac{m}{n} < 1$ on the set V of vertices. Let $S \subset V$ be a subset of vertices and assume that $|S| = \beta n$ with β small enough. Then, the induced subgraph $G_{V \setminus S}(n,m)$ is a random graph with the edges-vertices ratio c slightly less than m/n. Let

$$t(c) = \frac{1}{2c} \sum_{k=1}^{\infty} \frac{k^{k-1}}{k!} (2ce^{-2c})^k$$

and $1 - p_S(n)$ be the probability that the size of the largest component of $G_{V \setminus S}(n,m)$ is in the order of $(1-t(c))n$. The famous result on the size of the giant component in a random graph, see e.g. [30], indicates that $p_S(n)$ tends to zero. It is also true that $(1-t(c))$ is larger than 2/3 even for c well below 1. Notice that the probability for $G(n,m)$ to have a balanced partition of the form (S, A, B) is less than $p_S(n)$——the existence of such a balanced partition implies that the components of the induced subgraph $G_{V \setminus S}(n,m)$ are all of size less than $\frac{2}{3}n$. Since there are $\binom{n}{\beta n}$ such S, we could have shown that the threshold of having a linear treewidth is less than one if the probability $p_S(n)$ is exponentially small. Unfortunately, we currently do not know yet if such an exponential upper bound for $p_S(n)$ exists.

6.4 Treewidth of Random Models in AI and Computational Biology

In this section, we study the treewidth of the underlying graphs of several random models in AI and computational biology. As has been discussed at the beginning of this chapter (Section 6.1), the size of the treewidth characterizes the time and space complexities of several algorithms for these problems. A general conclusion to be drawn from the results of this section is that all of these algorithms will have an exponential behavior even for instances randomly generated well below the threshold of the solution-probability phase transition. It should be noted that instances randomly generated far from the phase transition have been known theoretically and/or empirically to be easy for backtracking algorithms.

6.4.1 Treewidth of Random CSPs

We consider the random CSP model $\mathcal{B}_{n,m}^{d,k,t}$, i.e., model B defined in Definition 3.2.6. Similar results hold for other random CSP models.

Recall that the *primal graph* of a CSP instance is a graph $G = G(V, E)$ where V corresponds to the set of variables X and $(v_i, v_j) \in E$ if and only if the corresponding variables x_i and x_j appear in some constraint at the same time. The correspondence between the primal graph of a CSP instance and the constraint hypergraph of the CSP instance is as follows: For each hyperedge (i.e. each constraint), make a clique on the set of vertices in the hyperedge. The primal graph is a *graph of cliques* whose edge set is the union of the edges of the cliques.

Theorem 6.4.1. *Let $G_{\mathcal{B}}(n, m)$ be the primal graph of the random CSP $\mathcal{B}_{n,m}^{d,k,t}$ and let $c^* = \frac{\log 2}{k \log 3 - \log(1+2^k)}$. Then, we have*

1. *if $\frac{m}{n} < \frac{1}{k(k-1)}$,*

$$\lim_n Pr\{tw(G_{\mathcal{B}}(n, m)) \leq k + 1\} = 1.$$

2. *if $\frac{m}{n} > c^*$, there is a constant $\delta > 0$ such that*

$$\lim_n Pr\{tw(G_{\mathcal{B}}(n, m)) > \delta n\} = 1.$$

Proof. For the case of $\frac{m}{n} < \frac{1}{k(k-1)}$, the constraint hypergraph $\mathcal{B}_{n,m}^{d,k,t}$ contains only hypertrees and unicycles **whp** (Lemma 2.2.1). It can be shown that the graph of cliques obtained from a k-homogenous hypergraph with only hypertrees and unicycles has a treewidth of at most $k + 1$.

The proof of the case $\frac{m}{n} > c^*$ is based on the same technique as that in the proof of Theorem 6.3.1. The only difference is that the primal graph $G_{\mathcal{B}}(n, m)$

is not the standard uniform random graph. Rather, $G_B(n, m)$ is a "graph of random cliques"—its edges are the union of the edges of m randomly selected cliques of size k. □

6.4.2 Treewidth of Random Bayesian Networks

Given a set of random variables $X = \{X_1, \cdots, X_n\}$, a Bayesian network is a pair $\mathcal{B}(G, P)$ where G is a directed acyclic graph over X and P defines a set of conditional probabilities $P_i = Pr\{X_i|pa(X_i)\}$ with $pa(X_i)$ being the parent of the node X_i. A Bayesian network provides a concise representation of the joint probability distribution of the random vector X. The *moral graph* of a Bayesian network is an undirected graph obtained by first connecting the parents of each node, and then changing all the directed edges into undirected ones.

One of the most important problems in Bayesian networks is inference, i.e., the problem of calculating the (conditional) probability for a subset of variables. There are three types of inference problems:

1. *probabilistic inference*, also called belief updating. The object is to compute the posterior probability of a subset of variables, given a subset of observed evidence variables;

2. *most probable explanation* (MPE). The task is to find a maximum probability instantiation consistent with a given set of observed evidence.

3. *maximum a posteriori* (MAP). The task is to find an instantiation to a subset of variables with maximum a posteriori probability conditional on a set of given evidence.

All of the three types of inference problems are NP-hard in general [40, 128], and hard to approximate up to a constant ratio [1, 45, 125].

For the class of singly-connected Bayesian networks, a polynomial propagation algorithm, called message passing, has been developed [123]. For general Bayesian networks, there are several algorithms for exact inference, including the tree-decomposition-based algorithm called the *tree-clustering or junction-tree* [123], the cycle-cutset (also called conditioning) algorithm [123], and the more general variable-elimination scheme called *bucket elimination* [50]. Similar algorithms have also been used in practical applications of CSPs [52, 81]. These algorithms' running times are exponential in the maximum size of the subsets in the tree-decomposition used by the algorithms [50, 51], which in turn is lower bounded by the treewidth of the underlying structures. The task of finding a tree-decomposition whose maximum subset size achieves the treewidth is known to be NP-hard [103], and many heuristics and approximation algorithms have been proposed in the literature of algorithmic graph theory and Bayesian networks [25, 28, 102, 103]. However, finding the best

tree-decomposition is far from resolving the fundamental complexity issues in these inference algorithms. It has been found that Bayesian networks usually have large treewidth [95]. Several heuristics have been proposed to achieve the time-space tradeoff, resulting in so-called *any-space* inference algorithms [47, 54].

As there has been growing interest in using randomly-generated instances to evaluate tree-decomposition based algorithms [98, 122], it is important to understand the typical behavior of the treewidth in random models of these problems so that experimental results can be properly interpreted. Unfortunately, there has been no generally accepted random models for Bayesain networks. In [111], random Bayesian networks are generated by converting random SAT instances in a way similar to the reduction in the NP-hardness proof [40]. In [89], random Bayesian networks are generated by using Markov chain Monte Carlo method to make sure the network is uniformly distributed. In [98, 122], random Bayesian networks are generated in a way similar to the classical random graphs with a restriction on the variable ordering so that the resulting networks are acyclic.

We define two random Bayesain network models and consider the typical size of the treewidth of their moral graphs.

Definition 6.4.1 (Random Bayesian Networks). *Given a set of random variables $X = (X_1, \cdots, X_n)$, a random Bayesian network $\mathcal{B}(n)$ is specified by selecting the parents of each node randomly and independently. If we assume that the node X_i chooses as its parent each of the rest of the nodes randomly and independently with the probability p_i, we use $\mathcal{B}(n, p_i, 1 \leq i \leq n)$ to denote the corresponding random model.*

Of course, the above random model is not guaranteed to generate directed acyclic graphs. To generate directed acyclic networks, we may consider a modified version of the model that first chooses a random order of the variables, and then let each variable select their parents only from the precedent variables according to the order. The idea of our analysis can be extended to this restricted model with some complication.

The second model that we will consider is simpler. It is defined on the set of directed acyclic bipartite graphs. A typical example of this type of Bayesian networks is the QMR-DT database where the upper layer has about 600 nodes representing diseases and the lower layer has about 4000 nodes representing the symptoms [95]. Even with such a simple structure, the exact inference generally remains intractable. See [95] for empirical evidence and [40] for an idea of an NP-complete proof.

Definition 6.4.2 (Random Bipartite Bayesian Networks). *A random bipartite Bayesian network $\mathcal{B}(V_1, V_2, k)$ is a Bayesian network in which V_1 and V_2 are respectively the sets of nodes of upper and lower layers, and each node $x \in V_2$ randomly chooses a set of k nodes in V_1 as its parents.*

For random bipartite Bayesian networks, we have the following

Theorem 6.4.2. *Let $\mathcal{B}(V_1, V_2, k)$ be a random bipartite Bayesian network with $|V_1| = n$, $|V_2| = m$. Let $tw(\mathcal{B}(V_1, V_2, k))$ denote the treewidth of the moral graph of $\mathcal{B}(V_1, V_2, k)$ and define $c^* = \frac{\ln 2}{k \ln 3 - \ln(1+2^k)}$. Then, we have*

1. *if $\frac{m}{n} < \frac{1}{k(k-1)}$,*

$$\lim_n \Pr\{\, tw(\mathcal{B}(V_1, V_2, k)) \leq k+1 \,\} = 1.$$

2. *if $\frac{m}{n} > c^*$, there is a constant $\delta > 0$ such that*

$$\lim_n \Pr\{\, tw(\mathcal{B}(V_1, V_2, k)) > \delta n \,\} = 1.$$

Proof. Let $G(V_1, V_2)$ be the moral graph of the Bayesian network and $G_1(V_1)$ be the induced graph of $G(V_1, V_2)$ on V_1. By the definition of the treewidth and the fact that $G(V_1, V_2)$ is bipartite, it can be shown that the treewidth of $G(V_1, V_2)$ is the maximum of $k+1$ and the treewidth of $G_1(V_1)$. The theorem is proved by applying Theorem 6.4.1 to the graph of random cliques $G_1(V_1)$. □

For general Bayesian networks, if we want to use theorem 6.4.1, then the cliques in their *moral graph* have to be added randomly and independently. This is however not an appropriate assumption in the context of Bayesian networks because (1) the generated networks are not guaranteed to be acyclic and (2) there is no reason to assume that each variable has the same constant number of parents. The random model introduced in Definition 6.4.1 is a first step toward a more realistic random model for Bayesian networks, where we assume that each node selects its parents randomly and independently. It should be noted that this model can still generate cyclic networks. However, the idea of the analysis on this model can be extended to more elaborated models with some complication.

Theorem 6.4.3. *Let $\mathcal{B}(n, p_i, 1 \leq i \leq n)$ be a random Bayesian network on n variables and $tw(n)$ the treewidth of its moral graph. Then, there exists a $0 < \delta < 1$ such that $\lim_n Pr\{tw(n) \leq \delta n\} = 0$ if $(\prod_{i=1}^{n}(1-p_i))^{\frac{1}{3}} < \frac{1}{2}$.*

Proof. Similar to the proof of theorem 6.4.1, let $\mathcal{P}$ be the set of all the $k-$partitions of the vertex set of the moral graph of the Bayesian network that satisfies the first two conditions of the definition of balanced partition. For a given $P = (S, A, B) \in \mathcal{P}$, let E be the event that P is a balanced partition, i.e., the event that there is no edge between vertices of A and vertices of B.

For each $1 \leq i \leq n$ with $X_i \in A$ (or $X_i \in B$), let E_i be the event that all of its parents are in $A \bigcup S$ (or in $B \bigcup S$ respectively). For $X_i \in S$, let E_i be the event that all of its parents are in $A \bigcup S$ or in $B \bigcup S$. We have

$$E = \bigcap_{1 \leq i \leq n} E_i.$$

Since by assumption, each node selects its parents independently from the others, we have

$$Pr\{E\} = \prod_{i=1}^{n} Pr\{E_i\}. \tag{6.6}$$

For $X_i \in A$ (or $X_i \in B$), we have

$$Pr\{E_i\} \leq (1-p_i)^{\frac{1}{3}(n-k-1)}$$

and for $X_i \in S$, we have

$$Pr\{E_i\} \leq 2(1-p_i)^{\frac{1}{3}(n-k-1)} - (1-p_i)^k.$$

The rest of proof is similar to that of theorem 6.4.1. □

Discussions

In the study of Bayesian network inference, randomly-generated networks have been used to evaluate and compare various inference algorithms [98, 122]. Our results show that the treewidth of the random instances is asymptotically in the order of the size of the networks even if the random model itself is quite sparse. This implies that purely random Bayesian networks are not adequate at least for the evaluation of tree-decomposition based exact inference algorithms. A natural question then is how to devise a random model that has a controlled treewidth. Motivated by the k-tree based definition of treewidth, we propose the following random model. Starting from a clique of k nodes, we add new nodes one at a time. The new node is then connected to the nodes of a randomly selected k-clique in the old graph. We illustrate the idea by giving the following random Bayesian network model.

Definition 6.4.3. *Let $X = (X_1, X_2, \cdots, X_n)$ be a random vector. A random Bayesian network with bounded treewidth(RBNBT) is a Bayesian network constructed using the following procedure*

1. *Randomly select k random variables and make the first $(k-1)$ of them parents of the kth variable;*
2. *Randomly select a variable X_i from the rest of the variables and a k-clique from the moral graph of the Bayesian network in the previous step. Make each variable of the selected k-clique a parent of X_i;*
3. *Repeat previous step until all the variables have been considered;*
4. *For each variable, randomly remove some variables from its parent set.*

It is easy to see that the moral graph of the RBNBT has a treewidth at most k with probability one for any problem size. It might be interesting to see how many parent variables we have to remove before the treewidth of the generated Bayesian network is strictly less than k.

6.4.3 Treewidth of NK Landscapes and Other Additive Fitness Functions

NK landscape, proposed by Kauffman [99], is a versatile model for the study of biological evolution and networks of biological molecules. An NK landscape consists of n variables $\{X_i, 1 \leq i \leq n\}$ and a set of control functions $\{f_i, 1 \leq i \leq n\}$. These variables represent the states of a set of genes, mRNAs, or proteins. In the context of biological networks, the control function f_i determines the next state of X_i; while in the context of biological evolution, f_i represents the fitness contribution of X_i to the overall fitness $f = \sum f_i$ of the whole genotype. In an NK landscape, the control function f_i depends on X_i and k other variables selected according to some rules.

In the study of gene networks, topics of current interest are (1) the behavior of the NK landscape as a dynamic system and (2) efficient methods to reconstruct the control functions that represent the interaction among genes. As a model for biological evolution, one of the major tasks is to characterize the relation between the degree of gene interactions, the shape of the fitness landscape, and the complexity of searching and exploring the landscapes to find genotypes with higher fitness. NK landscapes have also been widely used as a prototype and benchmark in the analysis of different genetic operators and the effects of different encoding methods on the performance of genetic algorithms [88, 94].

There are basically two classes of NK landscapes: *NK landscapes with adjacent neighborhood* and *NK landscapes with random neighborhood*. As an optimization problem, it is known that NK landscapes with adjacent neighborhood can be solved polynomially and NK landscapes with random neighborhood are usually NP-hard. On the other hand, the two classes of NK landscapes share almost identical statistical characteristics such as the average number of local minima and the average height of the local minima [138]. This has puzzled researchers in this field for a while. In fact, Weinberger speculated in the conclusion section of [138] that this might be related to the treewidth. In this section, we confirm Weinberger's speculation by proving that **whp** NK landscapes with random neighborhood have linear treewidth.

Definition 6.4.4. *An NK landscape*

$$f(x) = \sum_{i=1}^{n} f_i(x_i, \pi(x_i)), \tag{6.7}$$

is a real-valued function defined on binary strings of fixed length, where $n > 0$ is a positive integer and $x = (x_1, \cdots, x_n) \in \{0,1\}^n$. It is the sum of n local fitness functions f_i, $1 \leq i \leq n$. Each local fitness function $f_i(x_i, \pi(x_i))$ depends on the main variable x_i and a set $\pi(x_i)$ of k other variables called the neighbors of x_i.

1. *NK landscapes with random neighborhood. In this type of NK landscapes,*

$\pi(x_i)$ *consists of* k *variables randomly chosen from the set* $\{x_1, \cdots, x_n\} \setminus \{x_i\}$.

2. *NK landscapes with adjacent neighborhood. In this type of NK landscapes,* $\pi(x_i)$ *consists of the* k *variables with indices nearest to* i *(modulo* n*). To simplify the discussion, we assume in this paper that for each* i*,*

$$\pi(x_i) = (x_{max(0,i-\lceil \frac{k}{2} \rceil)}, \cdots, x_{i-1}, x_{i+1}, \cdots, x_{min(n,i+\lceil \frac{k}{2} \rceil)}). \tag{6.8}$$

In addition to NK landscapes, the following additive fitness functions are also widely used. It should be noticed these concepts are simply real-valued versions of some concepts related to random CSP instances.

Definition 6.4.5. *A function* $f : X = \{0,1\}^n \to [0, \infty)$ *is additive if it can be represented as a sum of lower dimensional functions*

$$f(x) = \sum_{C \in \mathcal{C}} f_C(x), \quad x = \{x_1, \ldots, x_n\} \in X,$$

where $\mathcal{C}$ *is a collection of subsets of* $\{x_1, \cdots, x_n\}$*. For each* $C \in \mathcal{C}$*,* $f_C(x)$ *only depends on the variables in* c*, and is thus called a local function. The order* k *of an additive function* f *is the size of the largest variable set in* $\mathcal{C}$*. Since we can always make the variable sets the same size by merging and/or adding dummy variables, we can assume that* $\mathcal{C}$ *consists of variable sets of size* k*.*

The interaction graph of the additive function is a graph $G_f = G_f(V, E)$ *where the vertex set* $V = \{x_1, \ldots, x_n\}$ *corresponds to the set of variables, and* $(x_i, x_j) \in E$ *if and only if there is a subset* $C \in \mathcal{C}$ *such that* $x_i \in C$ *and* $x_j \in C$*.*

The treewidth $tw(f)$ *of* f *is defined to be the treewidth of its interaction graph.*

For NK landscapes with adjacent neighborhood, we have the following result.

Theorem 6.4.4. *Let f(x) be an NK landscape with adjacent neighborhood. Then, we have* $k \leq tw(f) \leq 2k$*.*

Proof. Since the interaction graph f contains cliques of size $k+1$, its treewidth should be no less than k. We prove that $tw(f) \leq 2k$ by constructing a tree decomposition with a treewidth $2k$. Let $V = \{x_1, \cdots, x_n\}$ be the set of vertices, and let $V_0 = \{x_1, \cdots, x_k\}$. We construct $S = \{X_i, i \geq 1\}$, a collection of

subsets of the variables, as follows:

$$\begin{aligned}
X_1 &= \{x_1, \cdots, x_{k+1}\} \cup V_0, \\
X_2 &= \{x_2, \cdots, x_{k+2}\} \cup V_0, \\
&\cdots \\
X_{n-k} &= \{x_{n-k}, \cdots, x_n\} \cup V_0, \\
X_{n-k+1} &= \{x_{n-k+1}, \cdots, x_n, x_1\} \cup V_0, \\
X_{n-k+2} &= \{x_{n-k+2}, \cdots, x_n, x_1, x_2\} \cup V_0, \\
&\cdots \\
X_n &= \{x_n, x_1, x_2, \cdots, x_k\} \cup V_0,
\end{aligned}$$

and define a tree structure on S by assigning an edge between each of the pairs $(X_i, X_{i+1}), 1 \leq i \leq n-1$. It is easy to verify that the collection of subsets of variables and the tree structure specified in the above form a tree decomposition with a width $2k$. □

On the other hand, the following theorem states that NK landscapes with random neighborhood **whp** have linear treewidth.

Theorem 6.4.5. *Let f(x) be an NK landscape with random neighborhood. Then, for $k \geq 2$, there is a fixed constant $\delta > 0$ such that*

$$\lim_n Pr\{tw(f) \geq \delta n\} = 1. \tag{6.9}$$

Proof. The proof of the case $\frac{m}{n} > c^*$ is based on the same technique as that in the proof of Theorem 6.3.1. The only difference is that the interaction graph of f is neither the standard uniform random graph nor the "graph of random cliques". Instead, it consists of a set of n cliques $\{C_i, 1 \leq i \leq n\}$ where each C_i contains the vertex v_i and two other vertices uniformly selected from $\{v_1, \cdots, v_n\} \setminus \{x_i\}$. □

Chapter 7

Conclusions

In this thesis, we have explored several different aspects of the theoretical and empirical hardness of randomly-generated problem instances. For randomly-generated instances of constraint satisfaction problems, we identified a new class of algorithmically exploitable structures and proved that under certain distributions, random instances contain such structures with high probability (Chapter 4). In an effort to find a way to eliminate these structures from randomly-generated CSP instances, we established a connection between the famous notion of constraint consistency in the literature and the resolution complexity of random CSP instances. By embedding a recursive structure—the consistency core—into the distribution of the random CSP models, we proposed a novel scheme to generate random CSP instances with theoretically guaranteed resolution complexity and empirically confirmed hardness (Chapter 5). Our proposal resolved the long-standing problem of generating hard random CSP instances with bounded domain size that has troubled the society for several years.

While all of the results in Chapters 4 and 5 are aimed at backtracking search algorithms, we investigated in Chapter 6 the typical-case behavior of random instances in terms of search algorithms with a different flavor—those dynamic programming based algorithms whose time and space complexities are exponential in the treewidth of the underlying structures. We established an improved lower bound on the threshold for a graph to have a treewidth linear to the graph size. Similar techniques were then applied to random CSPs and Bayesian networks in AI and random fitness landscape models in computational biology and evolutionary computation. It was concluded that these dynamic programming based algorithms all have exponential behavior even on problem instances randomly-generated under a distribution that has been shown theoretically and/or empirically to be easy for backtracking search algorithms.

As for the implications of the current study to practice and the lessons learned from this study, I will try to discuss below several issues that have kept "puzzling" me throughout this research.

It is the Structure Plus the Algorithm

An important lesson learned from the study of the phase transition in NP-complete problems is the central importance of the structural information in a combinatorial search problem. It is now clearer why really hard problems "are well out of reach of any intelligent algorithms" —there is simply no small structural signature in these hard problems for any foreseeable intelligent algorithms to exploit [44]. This point of view is enhanced by the study in this thesis on the embedded easy subproblems and on the connection between constraint consistency and resolution complexity of randomly-generated CSP instances.

The enthusiasm in the phase transition of NP-complete problems stems from the close connection between the instance hardness and the phase transition of the solution probability. The study on the typical-case size of treewidth in Chapter 6 indicates that such a connection is indeed algorithm-dependent and is far from universal. For dynamic programming based algorithms that exploit quite different structural information, our study shows that contrary to some previous expectation, there is no hope to establish any kind of connections between the instance hardness and the solution probability phase transition—it is the phase transition of the size of the treewidth that plays the dominating role, and this phase transition occurs well before the solution probability transition.

Hard Instances as Benchmarks for Testing What?

The motivation to generate testing instances for algorithms and solvers in practice is, among others, to carry out one of the following tasks:

1. To find bugs in our implementation;
2. To study a random instance distribution itself of a given problem;
3. To identify the limitations of a specific class of algorithms; and
4. To look for principles regarding the design and the use of heuristics for tackling algorithmic problems.

Task 1 is the most basic and practical one, and has been an important topic in software engineering and software industry as exemplified by the adoption of the so-called *unit-testing paradigm* in the software development process. Interestingly, the theory of computation tells us that this is in fact a computationally unsovable task, and thus could not have been the original motivation of the study on the phase transition of NP-complete problems.

Task 2 is interesting and fun as it satisfies our curiosity about the unknown. But such practice should be avoided as much as possible in conducting empirical study of algorithms [93]. I have found out, however, that at least for me myself it is very tempting to do so.

It is thus clear that the whole purpose of generating hard instances is for the third and the fourth tasks—to study the limitations of specific class of algorithms and to draw guidance to the design of algorithms.

One of the original interests in the study of phase transitions in NP-complete problems is that by varying some distribution parameter, one can generate instances with a desired degree of hardness (for some specific class of algorithms). Unfortunately, perhaps largely motivated by the various solver competitions, there has been enthusiasm for generating various "hard" instances to blindly beat those solvers and algorithms. We can easily generate instances with a large number of cycles in their underlying graphs to fool the famous survey-propagation algorithm; we may come up with a random model whose instances are "hard" for any type of backtracking algorithms simply because no heuristic is going to work; or we can generate random CSPs or Bayesian networks based on standard random graphs in a straightforward way to evaluate (and easily beat) dynamic programming algorithms; and finally, there is the needle-in-the-haystack function which has no local minimum at all, but is surely extremely hard to optimize by any local search algorithms. What is difficult and non-trivial, however, is to devise random instances that reveal intrinsic connections between structures and the efficiency of algorithms and heuristics.

Where does theory meet practice?

A common criticism to theoretical analysis is that it is not practical. This is in some sense true because most of the theoretical results we can expect to get from current analytical techniques are **hopelessly** on the limiting behavior (See Figure 7.1 to find out how hopeless a theory could be). What scientists in practical fields really want is a theory that can explain phenomena occurring at a finite problem size, say n = 20, 100, or 1000.

On the other hand, we have theoretical physicists who use sophisticated mathematics, those that sometimes scare real mathematicians and theoretical computer scientists, to derive limiting results and to use these limiting behaviors to explain real-world phenomena. This methodology turns out to be very powerful and has been used for many years by physicists.

In the field of computer science, my impression is that the two tasks similar to those carried out by physicists are sometimes conducted separately by two groups of people. We have theoretical computer scientists and discrete mathematicians who have been publishing deep and elegant results but do not care much about interpreting real-world phenomena. On the other hand, we have computer scientists who have done excellent work in dealing with problems of practical size but do not have much belief in the potential intuitions that a theory on the limit behavior can provide.

I hope that this thesis helps in illustrating what a theory on the limit behavior can offer in practice. My experience in this study tells me that it

is not about the specific numbers or sizes that a theory can match—what really matters is the message and observation conveyed during the process of the theoretical analysis and the corresponding results. In the experiments carried out in Chapter 5, I never tried to find out how many flawed variables or embedded subproblems there could be in the instances generated from the two CSP models, the model B and the flawless model—the chance of finding one in problems of size n = 500 is not decently high. But still, it turns out that our proposed new model motivated by the theoretical analysis does make a real difference in terms of the instance hardness.

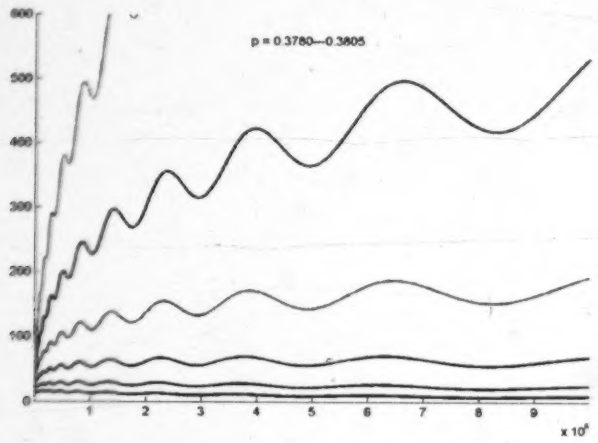

Figure 7.1: These curves are all supposed to drop to zero in the limit.

Bibliography

[1] A. Abdelbar and S. Hedetniemi. Approximating MAPs for belief networks is NP-hard and other theorems. *Artificial Inteligence*, 102:21–38, 1998.

[2] D. Achlioptas. *Threshold Phenomena in Random Graph Colouring and Satisfiability.* PhD thesis, Department of Computer Science, University of Toronto, Toronton, Canada, 1999.

[3] D. Achlioptas. A survey of lower bounds for random 3-SAT via differential equations. *Theoretical Computer Science*, 265(1-2):159–185, 2001.

[4] D. Achlioptas, P. Beame, and M. Molloy. A sharp threshold in proof complexity. In *ACM Symposium on Theory of Computing*, pages 337–346, 2001.

[5] D. Achlioptas and E. Friedgut. A sharp threshold for k-colorability. *Random Structures and Algorithms*, 14(1):63–70, 1999.

[6] D. Achlioptas, C. Gomes, H. Kautz, and B. Selman. Generating satisfiable problem instances. In *Seventeenth National Conference on Artificial Intelligence (AAAI'00)*, pages 256–261, 2000.

[7] D. Achlioptas, H. Jia, and C. Moore. Hiding satisfying assignments: Two are better than one. In *AAAI 2004*, pages 131–136.

[8] D. Achlioptas, L. Kirousis, E. Kranakis, D. Krizanc, M. Molloy, and Y. Stamation. Random constraint satisfaction: A more accurate picture. In *Proceedings of CP97*, pages 107–120. Springer, 1997.

[9] D. Achlioptas, L. Kirousis, E. Kranakis, and D. Krizane. Rigorous results for random (2+p)-SAT. *Theoretical Computer Science*, 265(1-2):109–129, 2001.

[10] D. Achlioptas and C. Moore. Almost all graphs with average degree 4 are 3-colorable. *Journal of Computer and System Sciences*, 67(2):441–471, 2003.

[11] D. Achlioptas and A. Naor. The two possible values of the chromatic number of a random graph. *Annals of Mathematics*, to appear.

[12] D. Achlioptas and Y. Peres. The random k-SAT threshold is $2^k log 2 - o(k)$. In *Proceedings of the 35th Annual Symposium on Theory of Computing, STOC03*, pages 223–231, 2003.

[13] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(47), 2002.

[14] N. Alon, J. H. Spencer, and P. Erdös. *The Probabilistic Method.* Wiley, 1992.

[15] Y. Asahiro, K. Iwama, and E. Miyano. Random generation of test instances with controlled attributes. In D. Johnson and M. Trick, editors, *Cliques, Colorings, and Satisfiability: DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 377–393. American Mathematical Society, 1996.

[16] A. Atserias, P. Kolaitis, and M. Vardi. Constraint propagation as a proof system. In *Proceedings of the 10th International Conference on Principle and Practice of Constraint Programming (CP'04)*, pages 77–91, 2004.

[17] F. R. Bach and M. I. Jordan. Thin junction trees. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, 2002.

[18] A. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results.* PhD thesis, University of Oregon, 1995.

[19] W. Barthel, A. Hartmann, M. Leone, F. Ricci-Tersenghi, M. Weigt, and R. Zecchina. Hiding solutions in random satisfiability problems: A statistical mechanics approach. *Phys. Rev. Lett.*, 88, 2002.

[20] R. Bayardo and R. Schrag. Using CSP look-back techniques to solve real world sat instances. In *Proc. of the 14th National Conf. on Artificial Intelligence*, pages 203–208, 1997.

[21] P. Beame, J. Culberosn, D. Mitchell, and C. Moore. The resolution complexity of random graph k-colorability. *Electronic Colloquium on Computational Complexity, TR04-012*, 2004.

[22] P. Beame, R. Karp, T. Pitassi, and M. Saks. The efficiency of resolution and Davis-Putnam procedures. *SIAM Journal on Computing*, 31(4):1048–1075, 2002.

[23] P. Beame, H. Kautz, and A. Sabharwal. On the power of clause learning. In *Proceedings of the 18th International Joint Conference in Artificial Intelligence (IJCAI)*, pages 94–99, 2003.

[24] P. Beame and T. Pitassi. Propositional proof complexity: Past, present, and future. *Bulletin of the European Association for Theoretical Computer Science*, 65:66–89, 1998.

[25] A. Becker and D. Geiger. A sufficiently fast algorithm for finding close to optimal junction trees. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 81–89. Morgan Kaufmann, 1996.

[26] E. Ben-Sasson. *Expansion in Proof Complexity.* PhD thesis, Department of Computer Science and Electrical Engineering, Hebrew University, 2001.

[27] E. Ben-Sasson and A. Wigderson. Short proofs are narrow - resolution made simple. *Journal of ACM*, 49(2), 2001.

[28] H. L. Bodlaender. A tourist guide through treewidth. Technical report, Technical Report RUU-CS-92-12, Department of Computer Science, Utrecht University, 1992.

[29] H. L. Bodlaender. Treewidth: algorithmic techniques and results. In *Lectures Notes in Computer Science 1295*, pages 19–36. Springer, 1997.

[30] B. Bollobas. *Random Graphs.* Cambridge University Press, 2001.

[31] M. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *Journal of Symbolic Logic*, 62(3):708–728, 1997.

[32] C. Borgs, J. Chayes, S. Mertens, and B. Pittel. Phase diagram for the constrained integer partitioning problem. *Random Structures and Algorithms*, pages 315–380, 2004.

[33] C. Borgs, J. Chayes, and B. Pittel. Phase transition and finite-size scaling for the integer partitioning problem. *Random Structures and Algorithms*, pages 247–288, 2001.

[34] V. Bouchitt and I. Todinca. Treewidth and minimum fill-in: Grouping the minimal separators. *SIAM Journal on Computing*, 31(1):212–232, 2001.

[35] P. Cheeseman, B. Kanefsky, and W. Taylor. Where the *really* hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 331–337. Morgan Kaufmann, 1991.

[36] M. Chvátal and B. Reed. Mick gets some (the odds are on his side). In *Proceedings of 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.

[37] V. Chvátal and E. Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*, 35(4):759–768, 1988.

[38] C. Coarfa, D. Demopoulos, Alfonso San Miguel Aguirre, D. Subramanian, and M. Vardi. Random 3-SAT: The plot thickens. In *Proceedings of the International Conference on Constraint Programming*, 2000.

[39] S. Cook and D. Mitchell. Finding hard instances of the satisfiability problem: A survey. In Du, Gu, and Pardalos, editors, *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1997.

[40] G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[41] J. Crawford and L. Auton. Experimental results on the crossover point in satisfiability problems. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93)*, pages 21–27. AAAI Press / The MIT Press, 1993.

[42] N. Creignou and H. Daud. Random generalized satisfiability problems. In *Fifth International Symposium on the Theory and Applications of Satisfiability Testing, SAT'02*, pages 17–26, 2002.

[43] J. Culberson, A. Beacham, and D. Papp. Hiding our colors. In *CP'95 Workshop, Studying and Solving Really Hard Problems*, pages 31–42, 21995.

[44] J. Culberson and I. Gent. Well out of reach: Why hard problems are hard. Technical Report APES-13-1999, APES Research Report, 1999.

[45] P. Dagum and M. Luby. Approximation probabilistic inference in bayesian belief networks is NP-hard. *Artificial Inteligence*, 60:141–153, 1993.

[46] V. Dalmau, P. Kolaitis, and M. Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proceedings Principles and Practices of Constraint Programming (CP-2002)*, pages 310–326. Springer, 2002.

[47] A. Darwiche. Recursive conditioning. *Artificial Intelligence*, 125:5–41, 2001.

[48] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[49] M. Davis and H. Putnam. A computing procedure for quantification. *Journal of the ACM*, 7:201–215, 1960.

[50] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113:41–85, 1999.

[51] R. Dechter and Y. Fattah. Topological parameters for time-space tradeoff. *Artificial Intelligence*, 125(1-2):93–118, 2001.

[52] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[53] R. Dechter and I. Rish. Directional resolution: the davis-putnam procedure, revisited. In *Proc. of the 4th International Conference on Knowledge Representation and Reasoning (KR'94)*, pages 134–145, 1994.

[54] R. Dechter and I. Rish. Mini-buckets: A general scheme for bounded inference. *Journal of ACM*, 50:107–153, 2003.

[55] H. Dixon, M. Ginsberg, and A. Parkes. Generalizing boolean satisfiability I: Background and survey of existing work. *Journal of Artificial Intelligence Research*, 21:193–243, 2004.

[56] S. Dorogovtsev and J. Mendes. Evolution of networks. *Advances in Physics*, 51:1079–1187, 2002.

[57] O. Dubois and Y. Boufkhad. A general upper bound for the satisfiability threshold of random r-SAT formulae. *J. of Algorithms*, 24:395–420, 1997.

[58] M. Dyer, A. Frieze, and M. Molloy. A probabilistic analysis of randomly generated binary constraint satisfaction problems. *Theoretical Computer Science*, 290:1815–1828, 2003.

[59] P. Erdös and A. Renyi. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, 5:17–61, 1960.

[60] J. Franco and A. Gelder. A perspective on certain polynomial-time solvable classes of satisfiability. *Discrete Applied Mathematics*, 125(2-3):177–214, 2003.

[61] J. Franco and M. Paul. Probabilistic analysis of the Davis-Putnam procedure for solving satisfiability. *Discrete Applied Mathematics*, 5:77–87, 1983.

[62] J. Franco and R. Swaminathan. Average case results for satisfiability algorithms under the random clause model. *Annals of Mathematics and Artificial Intelligence*, 20:357–391, 1997.

[63] J. Freeman. Hard random 3-sat problems and the davis-putman procedure. *Artificial Intelligence*, 81:183–198, 1996.

[64] E. C. Freuder. A sufficient condition for backtrack-free search. *Journal of the ACM*, 29(1):24–32, 1982.

[65] E. Friedgut. Sharp thresholds of graph properties, and the k-SAT problem. *J. Amer. Math. Soc.*, 12:1017–1054, 1999.

[66] A. Frieze and C. McDiarmid. Algorithmic theory of random graphs. *Random Structures and Algorithms*, 10:5–42, 1997.

[67] A. Frieze and M. Molloy. The satisfiability threshold for randomly generated binary constraint satisfaction problems. In *7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003*, pages 275–289, 2003.

[68] A. Frieze and B. Reed. Probabilistic analysis of algorithms. In H. Habib, C. McDiarmid, J. Ramirez, and B. Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 36–92. Springer-Verlag, 1998.

[69] Y. Gao. Phase transition of tractability in constraint satisfaction and Bayesian network inference. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-2003)*, pages 265–271. Morgan Kaufmann, 2003.

[70] Y. Gao and J. Culberson. An analysis of phase transition in NK landscapes. *Journal of Artificial Intelligence Research*, 17:309–332, 2002.

[71] Y. Gao and J. Culberson. On the treewidth of NK landscapes. In *Genetic and Evolutionary Computation Conference (GECCO-03)*, LNCS 2723, pages 848–954. Springer-Verlag, 2003.

[72] Y. Gao and J. Culberson. Resolution complexity of random constraint satisfaction problems: Another half of the story. In *LICS'03 Workshop on Typical Case Complexity and Phase Transitions*, 2003.

[73] Y. Gao and J. Culberson. Consistency and random constraint satisfaction models with a high constraint tightness. In *Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming (CP-2004)*, pages 17–31, 2004.

[74] I. Gent, H. Hoos, P. Prosser, and T. Walsh. Morphing: Combining structure and randomness. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, pages 654–660, Orlando, Florida, 1999.

[75] I. Gent, E. MacIntyre, P. Prosser, B. Smith, and T. Walsh. Random constraint satisfaction: Flaws and structure. *Constraints*, 6(4):345–372, 2001.

[76] I. Gent and T. Walsh. Analysis of heuristics for number partitioning. *Computational Intelligence*, 14(3):430–451, 1998.

[77] A. Goerdt. A threshold for unsatisfiability. *,Journal of Computer and System Sciences 33 (1996) 469–486*, 53(3):469–486, 1996.

[78] A. Goldberg, P. Purdom, and C. Brown. Average time analysis of simplified Davis-Putname procedures. *Information Processing Letter*, 15:72–75, 1982.

[79] C. Gomes and B. Selman. Algorithm portfolios. *Artificial Intelligence*, 126:43–62, 2001.

[80] C. Gomes and D. Shmoys. Completing quasigroups or latin squares: A structured graph coloring problem. In *Proceedings of the Computational Symposium on Graph Coloring and Extensions*, 2002.

[81] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural csp decomposition methods. *Articial Intelligence*, 124(2):243–282, 2000.

[82] J. Gu, P. Purdom, J. Franco, and B. Wah. Algorithms for satisfiability (sat) problem: A survey. In *Discrete Mathematics and Theoretical Computer Science: Satisfiability (SAT) Problem*, pages 19–152. American Mathematical Society, 1997.

[83] A. Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.

[84] H. Hatami and M. Molloy. Sharp thresholds for constraint satisfaction problems and homomorphisms. *submitted*, 2004.

[85] Brian Hayes. Can't get no satisfaction. *American Scientist*, 85(2):108–112, 1997.

[86] Brian Hayes. The easiest hard problem. *American Scientist*, 90(2):113–117, 2002.

[87] Brian Hayes. On the threshold. *American Scientist*, 91(1):12–17, 2003.

[88] W. Hordijk. A measure of landscapes. *Evolutionary Computation*, 4(4):335–360, 1997.

[89] J. IDE and F. Cozman. Generating random bayesian networks. In *Brazilian Symposium on Artificial Intelligence*. MIT Press, 2002.

[90] G. Istrate. Phase transitions and all that. Technical report, 2002.

[91] S. Janson. Large deviations for sums of partly dependent random variables. *Random Structures and Algorithms*, 24(3):234–248, 2004.

[92] F. V. Jensen and F. Jensen. Optimal junction trees. In R.L. de Mantaras and D. Poole, editors, *Proceedings of the 10 th conference on uncertainty in artificial intelligence*, pages 360–366. Morgan Kaufmann, 1994.

[93] D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, pages 215–250. American Mathematical Society, 2002.

[94] T. C. Jones. *Evolutionary Algorithms, Fitness Landscapes and Search.* PhD thesis, University of New Mexico, Albuquerque, NM, 1995.

[95] M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[96] D. Karger and N. Srebro. Learning markov networks: Maximum bounded tree-width graphs. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, pages 392–401, 2001.

[97] M. Karonski and T. Luczak. The phase transition in a random hypergraph. *J. Comput. Appl. Math.*, 142:125–135, 2002.

[98] K. Kask and R. Dechter. Stochastic local search for bayesian networks. In *Workshop on AI and Statistics (AI-STAT-99)*, pages 113–122, 1999.

[99] S. Kauffman. *The Origins of Order: Self-organization and Selection in Evolution.* Oxford University Press, Inc., 1993.

[100] J. H. Kim and V. H. Wu. Sandwiching random graphs. *Advances in Mathematics, to appear*, 2002.

[101] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.

[102] U. Kjærulff. Optimal decomposition of probabilistic networks by simulated annealing. *Statistics and Computing*, 2:7–17, 1991.

[103] T. Kloks. *Treewidth: Computations and Approximations.* Springer-Verlag, 1994.

[104] M. Krivelevich. Coloring random graphs - an algorithmic perspective. In *Proceedings of the 2nd Colloquium on Mathematics and Computer Science (MathInfo'2002)*, pages 175–195, 2002.

[105] M. Krivelevich, B. Sudakov, and V. H. Vu. A sharp threshold for network reliability. *Combinatorics, Probability and Computing*, 11:465–474, 2002.

[106] V. Kumar. Algorithms for constraint satisfaction: A survey. *AI Magazine*, 13(1):32–44, 1992.

[107] Leonid A. Levin. Average case complete problems. *SIAM J. Comput.*, 15(1):285–286, 1986.

[108] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[109] O. Martin, R. Monasson, and R. Zecchina. Statistical mechanics methods and phase transition in optimization problems. *Theoretical Computer Science*, 265:3–67, 2001.

[110] C. McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics*, London Mathematical Society Lecture Note Series, vol. 141, pages 148–188. Cambridge Univ. Press, 1989.

[111] O. J. Mengshoel, D. Roth, and D. C. Wilkins. Hard and easy Bayesian networks for computing the most probable explanation. Technical report, UIUCDCS-R-2000-2147, 2000.

[112] D. Mitchell. *The Resolution Complexity of Constraint Satisfaction.* PhD thesis, Department of Computer Science, University of Toronto, Canada, 2002.

[113] D. Mitchell. Resolution complexity of random constraints. In *Proceedings Principles and Practices of Constraint Programming (CP-2002)*, pages 295–309. Springer, 2002.

[114] D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of SAT problems. In *Proceedings of the 10th Natl. Conf on Artificial Intelligence*, pages 459–465. AAAI Press, 1992.

[115] M. Molloy. Models and thresholds for random constraint satisfaction problems. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 209 – 217. ACM Press, 2002.

[116] M. Molloy and M. Salavatipour. The resolution complexity of random constraint satisfaction problems. In *Proceedings of FOCS 2003*, 2003.

[117] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. 2+p-sat: Relation of typical-case complexity to the nature of the phase transition. *Random Structure and Algorithms*, 15:414, 1999.

[118] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transitions'. *Nature*, 400, 1999.

[119] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science*, 7:95–132, 1974.

[120] E. M. Palmer. *Graphical Evolution.* John Wiley Sons, 1985.

[121] G. Pan and M. Vardi. Search vs. symbolic techniques in satisfiability solving. In *The Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT 2004)*, 2004.

[122] J. Park and A. Darwiche. Approximating MAP using stochastic local search. In *UAI01*, 2001.

[123] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kauffmann, 1988.

[124] B. Pittel, J. Spencer, and N. Wormald. Sudden emergence of a giant k-core in a random graph. *Journal of Combinatorial Theory (B)*, pages 111–151, 1996.

[125] D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

[126] L. Sanchis. Generating hard and diverse test sets for NP-hard graph problems. *Discrete Applied Mathematics*, 58:35–66, 1995.

[127] B. Selman, D. Mitchell, and H. Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81(1-2):17–29, 1996.

[128] S. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Inteligence*, 68:399–410, 1994.

[129] B. Smith. Constructing an asymptotic phase transition in random binary constraint satisfaction problems. *Theoretical Computer Science*, 265(1-2):265–283, 2001.

[130] P. Stadler, W. Hordijk, and J. Fontanari. Phase transition and landscape statistics of the number partitioning problem. Technical Report SFI working paper 03-02-006, Santa Fe Institute, 2003.

[131] J. M. Steele. *Probability Theory and Combinatorial Optimization*. NSF-CBMS Volume 69, Society for Industrial and Applied Mathematics, 1997.

[132] C. Subramanian, M. Furer, and C. Madhavan. Algorithms for coloring semi-random graphs. *Random Structures and Algorithms*, pages 125–158, 1998.

[133] A. Urquhart. The complexity of propositional proofs. *The Bulletin of Symbolic Logic*, 1(4):425–467, 1995.

[134] B. Vandegriend and J. Culberson. The $G_{n,m}$ phase transition is not hard for the Hamiltonian Cycle problem. *Journal of Artificial Intelligence Research*, 9:219–245, 1998.

[135] V. H. Vu. Concentration of non-lipschitz functions and applications. *Random Structures and Algorithms*, 20(3):262–316, 2002.

[136] T. Walsh. Search in a small world. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1172–1177, 1999.

[137] T. Walsh. Search on high degree graphs. In *Proceedings of IJCAI-2001*, 2001.

[138] E. D. Weinberger. NP completeness of Kauffman's NK model, a tunable rugged fitness landscape. Technical Report Working Papers 96-02-003, Santa Fe Institute, Santa Fe, 1996.

[139] N. C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics*, London Mathematical Society Lecture Note Series, vol. 276, pages 239–298. Cambridge Univ. Press, 1999.

[140] K. Xu and W. Li. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 12:93–103, 2000.

[141] L. Zhang, C. Madigan, M. Moskewicz, and S. Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *Proceedings of International Conference on Computer Aided Design (ICCAD2001)*, 2001.

[142] L. Zhang and S. Malik. The quest for efficient boolean satisfiability solvers. In *Proceedings of 8th International Conference on Computer Aided Deduction(CADE 2002)*, 2002.